Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно–Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2017г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.–м.н.,
доцент
_____/А.А.Замышляева
«____»_____2017 г.

Разработка модуля «Биржа труда для студентов» сайта факультета
Математики, механики и компьютерных технологий.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.03.04.2017.065.ПЗ ВКР

Руководитель работы, старший преподаватель
_____/М.Ю. Сартасова
«____»_____2017 г.
Автор работы
Студент группы ЕТ–484
_____/ А.Н. Филиппов
«____»_____2017 г.
Нормоконтролер, доцент
_____/Т.Ю. Оленчикова
«____»_____2017 г.

Челябинск 2017

АННОТАЦИЯ

Филиппов А.Н. Разработка модуля «Биржа труда для студентов» сайта факультета Математики, механики и компьютерных технологий.– Челябинск: ЮУрГУ, ЕТ–484, 54 с., 47 ил.,12 табл., библиогр. список – 21 наим., 1 прил.

В данной работе реализована информационная система «Биржа труда для студентов» сайта факультета Математики, механики и компьютерных технологий.

Выполнен обзор существующих сайтов для поиска работы. Рассмотрены существующие технические решения для создания сайта и среды разработки. Разработан пользовательский интерфейс, удовлетворяющий требованиям простоты и минимальных знаний о структуре сайта.

В ходе работы была спроектирована база данных, разработана архитектура системы. Программная реализация осуществлена на языке PHP с использованием СУБД MySQL. В приложении приведен исходный код системы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время одним из популярных онлайн-сервисов являются системы для поиска работы.

Выпускникам высших учебных заведений в большинстве случаев сложно найти работу, соответствующую их специальности. Причиной этого чаще всего является требование наличия опыта работы или знания технологий, не известных выпускнику. Кроме этого, при поиске через интернет-системы есть вероятность того, что работодатель может оказаться ненадёжным. Поэтому выпускнику выгоднее устраиваться на работу через высшее учебное заведение.

При выборе бесплатного сервиса по поиску сотрудников предприниматели существенно сокращают свои расходы, но могут получить специалиста не совсем той квалификации, которая им требуется. В наше время, информационная система трудоустройства выпускников вуза станет отличным инструментом для скорейшей реализации как возможностей студентов, так и запросов предпринимателей.

Целью данной работы является разработка модуля «Биржа труда для студентов» для сайта факультета Математики, механики и компьютерных технологий.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) выполнить обзор существующих сайтов для поиска работы;
2) провести анализ существующих технологий создания сайтов;
3) выбрать язык и платформу проектирования;
4) разработать архитектуру системы;
5) разработать базу данных;
6) разработать и реализовать алгоритмы системы;
7) выполнить проверку работы системы.

1 ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СПОСОБОВ СОЗДАНИЯ WEB–ПРИЛОЖЕНИЙ

1.1 Существующие информационные системы для поиска работы

1.1.1 Карьера студентов факультета ММиКТ

Модуль является частью сайта факультета и доступен по адресу http://math.susu.ru/index.php/studentam/karera–testovyj–rezhim [1] (рисунок 1.1).

Рисунок 1.1 – Главная страница модуля «Карьера студентов факультета ММиКТ»

На главной странице размещено число вакансий, резюме, зарегистрированных компаний. Сверху расположено главное меню, содержащее меню "Вакансии","Резюме" слева и кнопки регистрации и входа справа (рисунок 1.2).

Рисунок 1.2 – Главное меню

Меню вакансий и резюме содержат пункты добавления, просмотра списка и поиска. Все пункты доступны всем типам пользователей, что является недоработкой интерфейса.

Для размещения вакансий или резюме необходимо зарегистрироваться, причём самостоятельная регистрация предусмотрена только для компаний (рисунок 1.3). Студентов может регистрировать только администратор.

На странице резюме работодатель может связаться с интересующим его студентом по электронной почте, нажав на соответствующую кнопку и заполнив форму.

Рисунок 1.3 – Регистрация работодателя

Для создания вакансии или резюме необходимо заполнить соответствующие формы (рисунок 1.4).

Рисунок 1.4 – Форма добавления вакансии

Уникальной особенностью сайта является система навыков, разделённых по категориям, например, «Языки программирования». Для указания навыков в резюме или вакансии их нужно перетащить из меню в поле формы. Пользователь при создании или редактировании резюме или вакансии может добавить новые навыки, которые помещаются в категорию «Без категории и новые». Редактировать навыки и категории может только администратор.

Поиск, как резюме, так и вакансий осуществляется только по навыкам (рисунок 1.5).

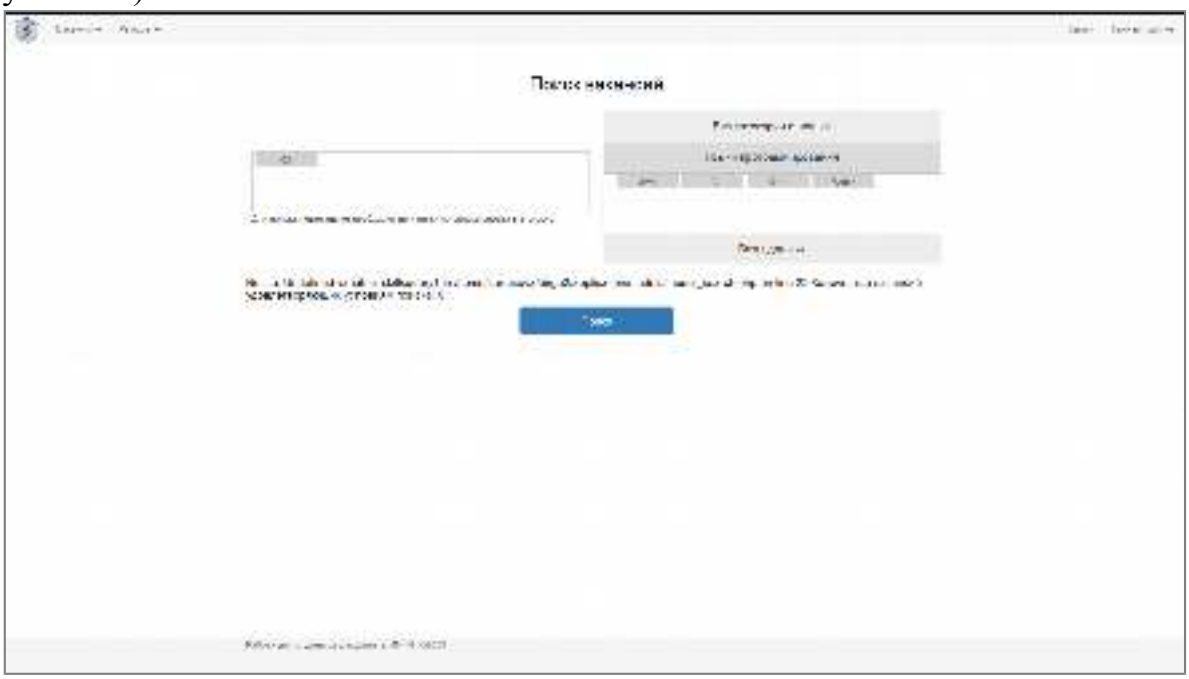

Рисунок 1.5 – Поиск вакансий по навыкам

Система навыков удобна для поиска, но неудобна при составлении вакансии или резюме с точки зрения пользователя. Например, в случае, если требуется указать уровень владения навыком или какое–то требование, необходимое только одному работодателю. Другими словами, отсутствует возможность формулировать требования в свободной форме.

Пользователь в личном кабинете может изменить свои данные и пароль. В личном кабинете компании перечислены её текущие вакансии.

Достоинства модуля:

- простота интерфейса;

Недостатки модуля:

- недоработки интерфейса;
- ограниченность поиска;
- невозможность свободно формулировать требования;
- на страницу компании можно перейти только из вакансии
- невозможность самостоятельной регистрации студентов.

1.1.2 FutureToday

Сайт доступен по адресу http://fut.ru [2] (рисунок 1.6).

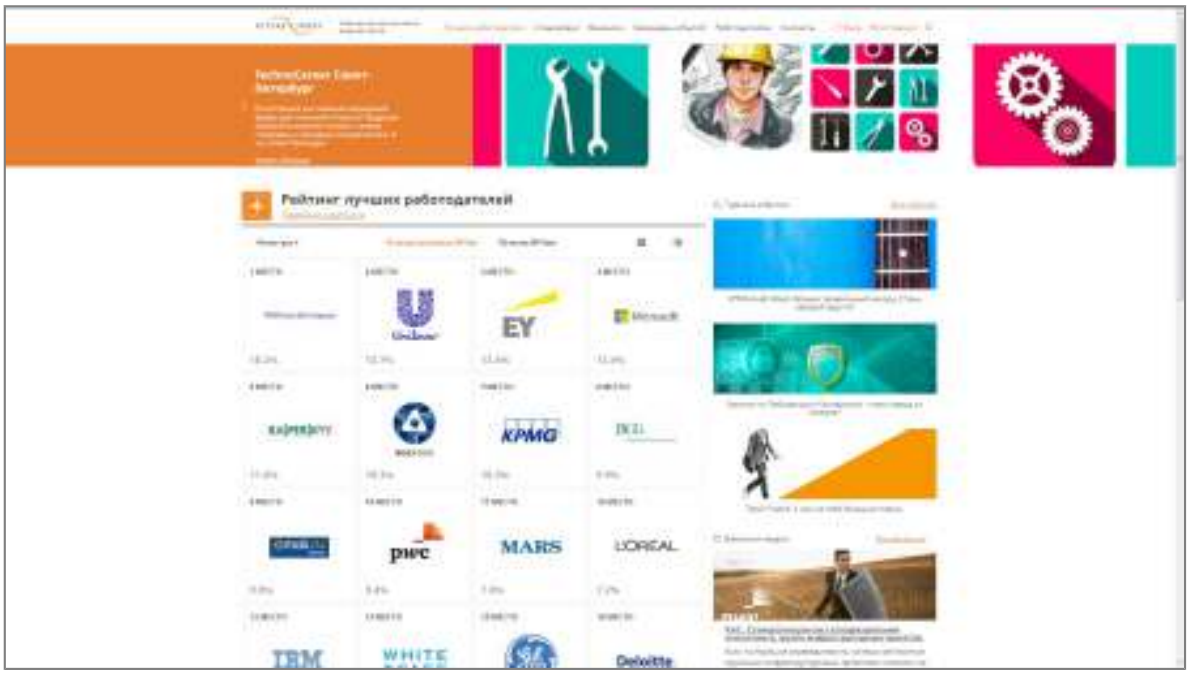

Рисунок 1.6 – Главная страница сайта «FutureToday»

Для размещения резюме или вакансии необходимо зарегистрироваться.

На сайте присутствует рейтинг компаний из разных отраслей, интервью со стажёрами и соискателями, уже принятыми на работу. Есть каталог компаний, разделённый по отраслям (рисунок 1.7).

Рисунок 1.7 – Каталог компаний

В разделе «Стажировки» присутствует информация о доступных стажировках, разделённых по отраслям, как оплачиваемых, так и неоплачиваемых (рисунок 1.8).

Рисунок 1.8 – Стажировки

В разделе «События» собрана информация о наборах, семинарах, днях открытых дверей, конкурсах и других мероприятиях, проводимых компаниями (рисунок 1.9).

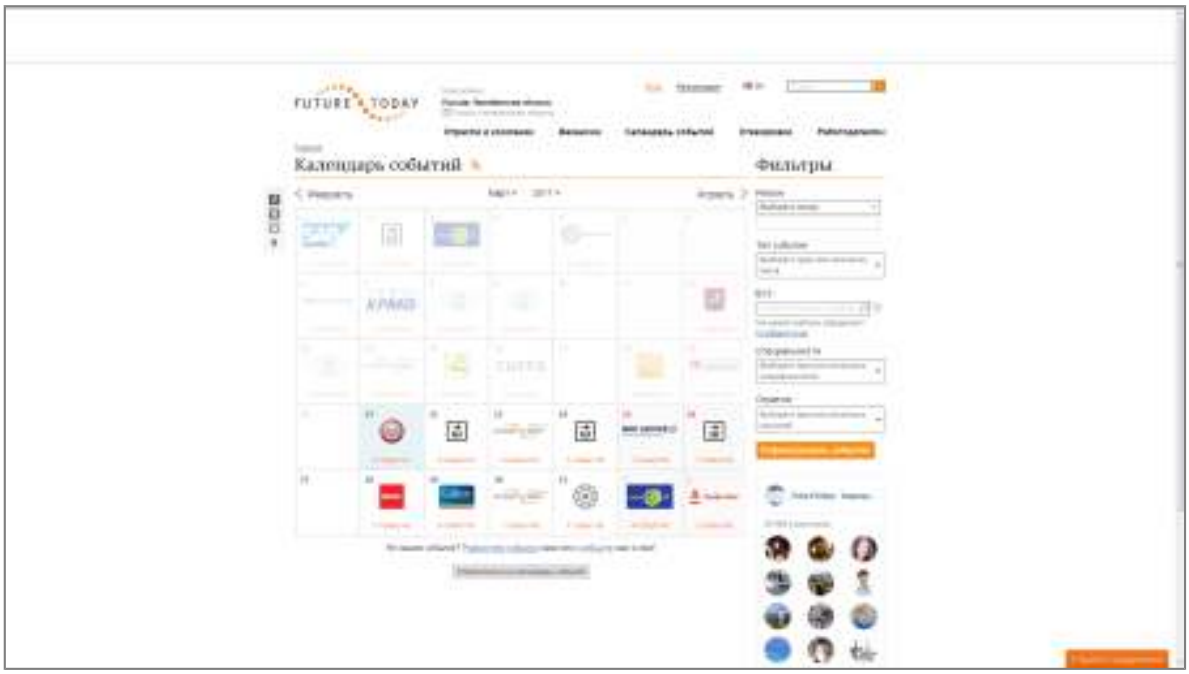

Рисунок 1.9 – События

Присутствует рейтинг работодателей, но далеко не у всех есть вакансии на сайте.

Компания может вести блог на сайте и отвечать на вопросы соискателей, а также рассылать им по электронной почте информацию о событиях и вакансиях (рисунок 1.10).

Рисунок 1.10 – Блог компании

Незарегистрированному пользователю доступен поиск только по вакансиям, компаниям и событиям (рисунок 1.11). Автоматически определяется регион пользователя и предлагается поиск вакансий только в нём.

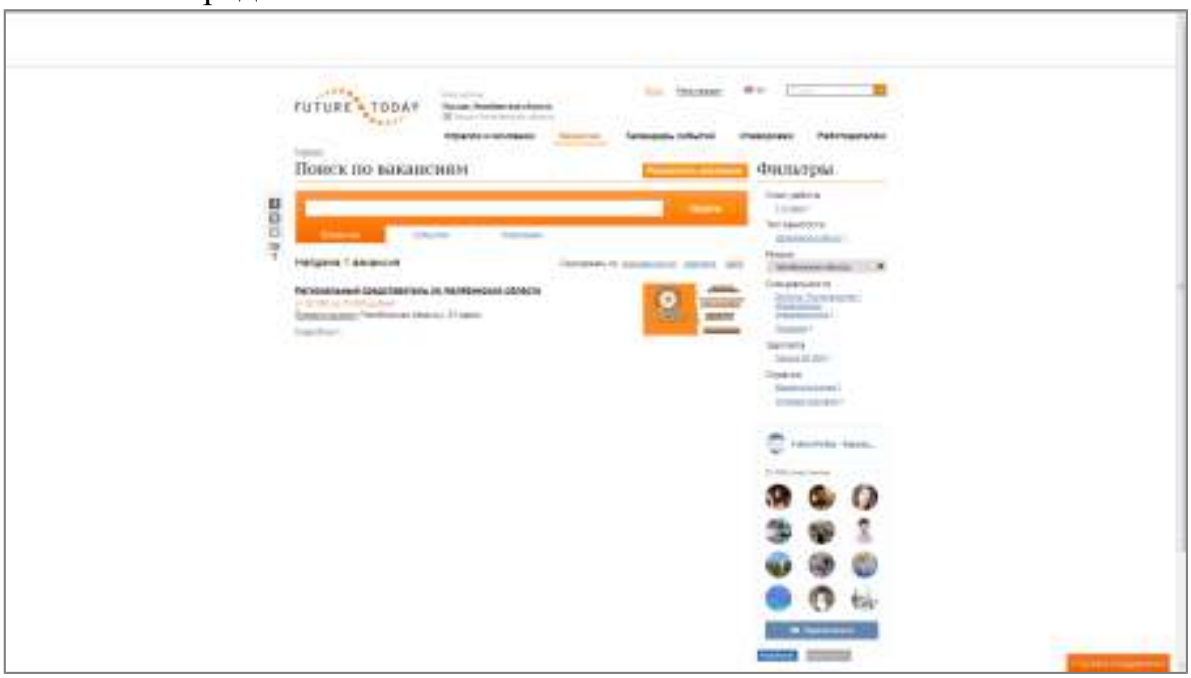

Рисунок 1.11 – Поиск вакансий

При поиске вакансий можно применить фильтры по опыту работы, типу занятости, специальности, зарплате и отрасли.

Также на сайте может быть зарегистрирован ВУЗ. ВУЗу предоставляются те же возможности, что и обычному пользователю: общение студентов с работодателями, новости, рассылка (определённой группе студентов, например, обучающихся на последнем курсе), а также автоматическое добавление вакансий, предназначенных студентам ВУЗа.

Достоинства:

- подробная фильтрация поиска;
- блоги компаний;
- возможность регистрации ВУЗа.

Недостатки:

- некоторая перегруженность интерфейса.

1.1.3 Зарплата.ру

Сайт доступен по адресу http://chelyabinsk.zarplata.ru/?gm [3].
Сайт предназначен для поиска работы в Челябинске и Челябинской области.
Главная страница разделена на боли, идущие сверху вниз (рисунок 1.12).

Рисунок 1.12 – Главная страница

Сверху расположено главное меню, далее идёт поиск вакансий, ниже вакансии, разделённые по отраслям. Отдельно выделены временная работа и удалённая работа. Далее идёт блок "Работодатели дня", "Вакансии дня", статьи, посвящённые трудоустройству и работе и блок новостей.

При поиске могут быть применены дополнительные фильтры, которые становятся доступны после нажатия на кнопку "Ещё фильтры" (рисунок 1.13).

Рисунок 1.13 – Дополнительные фильтры поиска

На странице вакансий присутствует тот же поиск, что и на главной странице. Слева располагается список вакансий, а справа – рубрика "Один день в компании" и список работодателей дня.

Список вакансий может быть представлен в виде, собственно, списка (рисунок 1.14), так и в виде отметок на карте Яндекса (рисунок 1.15).

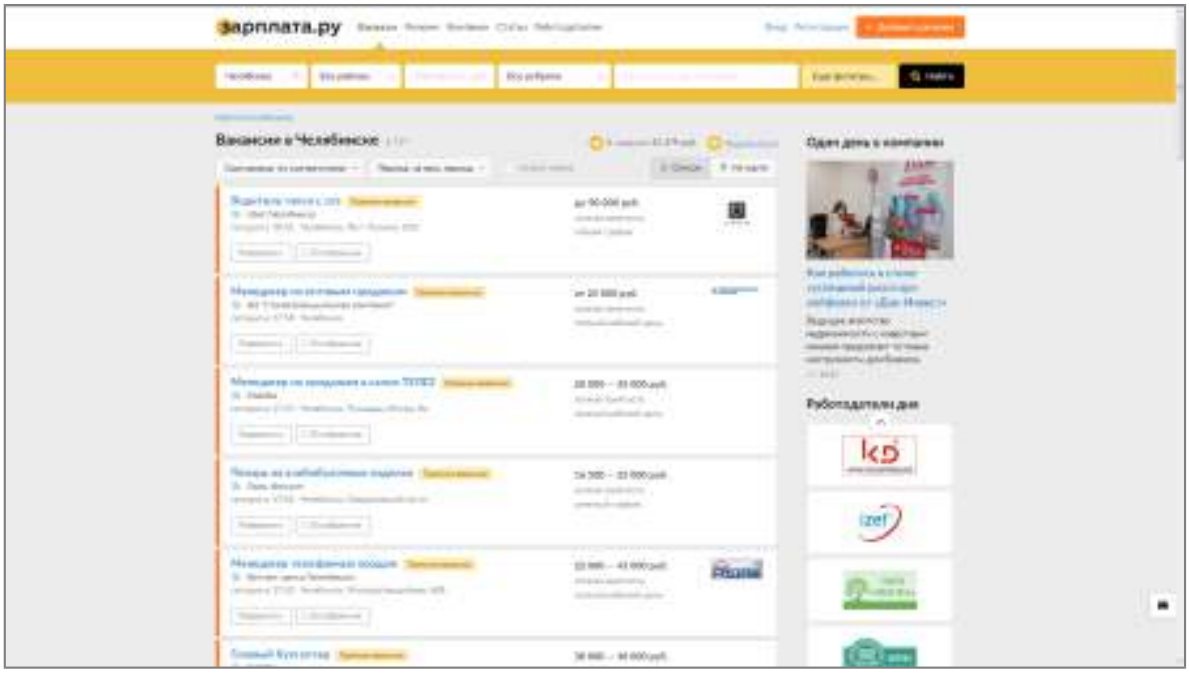

Рисунок 1.14 – Список вакансий

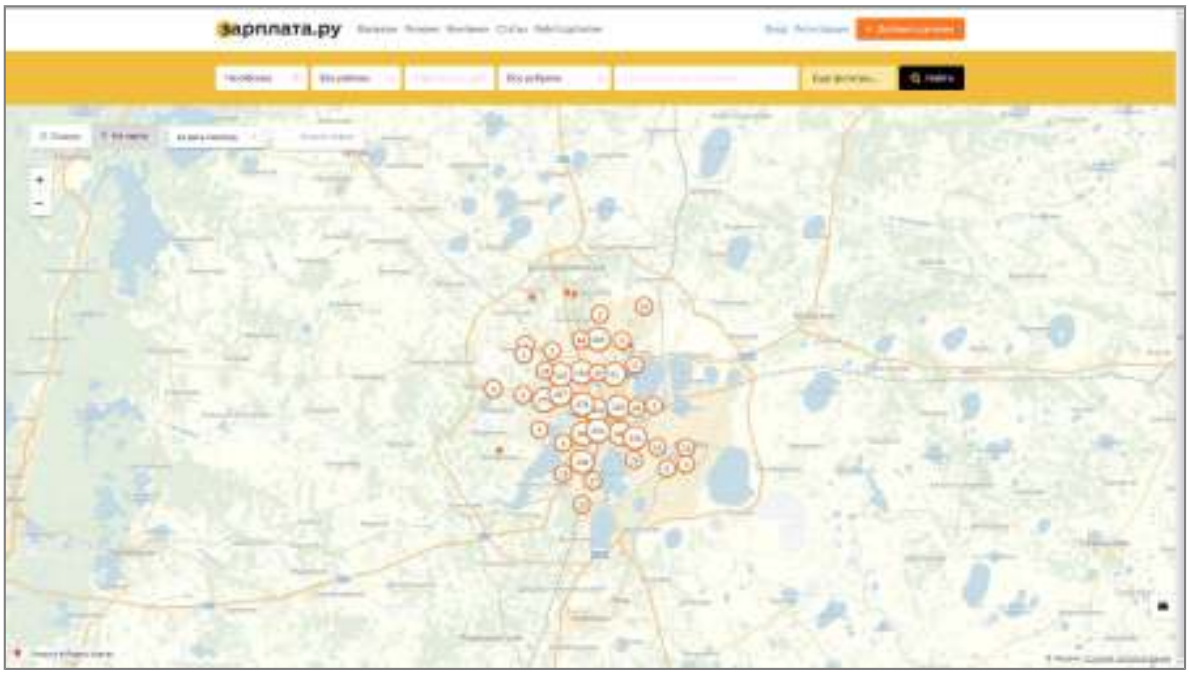

Рисунок 1.15 – Карта вакансий

Просмотр списка резюме доступен всем пользователям, но для их подробный просмотр доступен только работодателям.

На странице компании, на которую пользователь может перейти из списка компаний, содержится краткая информация о ней, контакты, список вакансий и карта филиалов (рисунок 1.16).

Рисунок 1.16 – Страница компании

Также на сайте публикуются статьи, новости и фоторепортажи, посвящённые работе и карьере (рисунок 1.17).

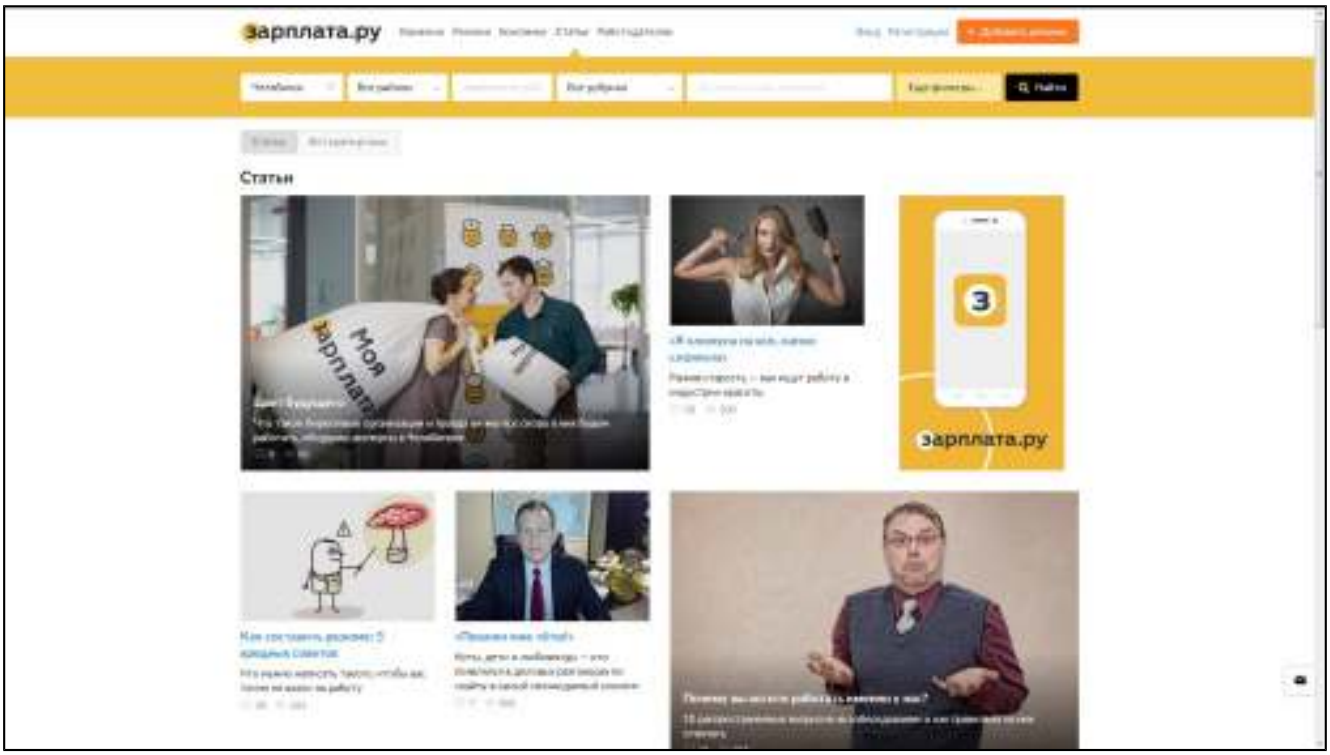

Рисунок 1.17 – Статьи и фоторепортажи о работе

Достоинства:

- основные функции доступны на главной странице;
- отображение вакансий на карте;

Недостатки:

- список резюме доступен всем пользователям.

1.2 Базовые технологии разработки сайтов

1.2.1 Статические сайты

Статический сайт состоит из HTML-страниц, которые никак не изменяются во время работы пользователя.

Страницы сайта данного типа создаются либо вручную, либо в специальных редакторах. Единственным способом обновления информации на статических страницах является прямое изменение HTML кода.

Достоинства статических сайтов:

- нагрузка на сервер является минимальной;
- быстрая загрузка;
- низкая стоимость разработки;
- переносимость не составляет труда.

Недостатки:

- сложность обновления информации и внесения изменений;
- управлять таким сайтом может только специалист в области разработки веб-приложений;
- сложность поддержки резко возрастает при увеличении количества страниц[4].

В настоящее время статические страницы чаще всего используются в качестве частей динамических сайтов.

1.2.2 Динамические сайты

Страницы динамического сайта формируется на стороне сервера частично или полностью, и, как вариант, составляются из существующей страницы при изменении данных пользователем.

Динамические страницы формируются на основе некоего шаблона. Данные для страницы чаще всего собираются на основе результатов запросов к базе данных.

Когда страница запрашивается пользователем, необходимая информация размещается в соответствующем месте в шаблоне, образуя готовую страницу, и затем пересылается в браузер, который в свою очередь отображает ее.

Таким образом, для того, чтобы обновить содержимое страницы необходимо только лишь изменять соответствующую информацию в базе данных. Информация в базе данных может быть изменена как непосредственно в СУБД, так и посредством взаимодействий пользователя со страницами сайта.

Преимущества динамических сайтов:

- все страницы динамического сайта могут быть сгенерированы на основе небольшого количества шаблонных страниц;
- поддерживать сайт может пользователь, не знакомый с разработкой веб-приложений;
- информация на странице изменяется, реагируя на действия пользователя.

Недостатки динамических сайтов:

- необходимо постоянно оптимизировать код для лучшей работы на различных серверах;
- высокая стоимость создания и поддержки [4].

В настоящее время имеет смысл создавать именно динамические сайты, так как чисто статические сайты уже не соответствуют требованиям современного мира.

1.3 Языки программирования

1.3.1 HTML и CSS

HTML — стандартизированный язык разметки документов в сети Интернет. Большинство веб–страниц содержат описание разметки на языке HTML или XHTML. HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства[5].

Язык XHTML является более строгим вариантом HTML, он следует всем ограничениям XML и, по сути, является приложением языка XML к области разметки гипертекста.

В сети Интернет HTML–страницы, как правило, передаются браузерам от сервера по протоколам HTTP в виде простого текста или HTTPS, с использованием шифрования.

CSS, или каскадные таблицы стилей, используются для задания оформления HTML или XML страниц. Стили задаются как для тегов, так и для элементов с конкретными идентификаторами, классами или именами[6].

В HTML5 появились новые тэги для создания структуры сайта и его содержимого, что избавляет разработчика от необходимости использовать дополнительные плагины или особую разметку. Добавлены стандартные элементы, присутствующие на большинстве современных сайтов: заголовок, завершитель, боковые и навигационные панели, статьи.

Многие элементы HTML5 были упрощены, и для них были введены значения по умолчанию, например в теге <script> не нужно указывать язык, если используется JavaScript.

Также улучшению подверглись и формы. Добавлены в качестве стандартных элементов ползунки, календари и диалоги выбора цвета.

В CSS3 появились расширенные селекторы, графические усовершенствования и улучшенная работа со шрифтами. Селекторы в CSS3 позволяют выделять отдельные элементы, такие как чётные и нечетные строки в таблице или последний абзац в группе.

В HTML5 стили могут быть заданы только в CSS, использование стилдей непосредственно в коде HTML запрещено.

Расширены возможности визуального оформления компонентов страниц: тени, градиенты, деформации и повороты.

HTML5 позволяет воспроизводить аудио и видео без использования сторонних технологий.

Спецификации еще не утверждены окончательно и все в них может измениться. Также не все браузеры поддерживают новые элементы интерфейса, введённые в HTML5 [7].

1.3.2 PHP

PHP (расшифровывается как PHP: Hypertext Preprocessor) – скриптовый язык программирования общего назначения с открытым исходным кодом. Синтаксис основан на C, Perl и Java. Применяется во множестве интернет–проектов. начиная от небольших личных страниц и заканчивая сайтами крупных корпораций[8].

PHP позволяет создавать динамически генерируемые страницы. Код на нём может встраиваться непосредственно в HTML между специальным открывающим тегом <?php и закрывающим тегом ?>. Файлы имеют расширение .php.

Преимущества языка PHP:

- является свободным программным обеспечением, распространяемым под собственной лицензией – PHP license;
- достаточно легок в освоении;
- обладает обширным сообществом пользователей и разработчиков;
- имеет развитую поддержку баз данных;

- имеется огромное количество библиотек и расширений языка;
- может использоваться в изолированной среде;
- имеет встроенные средства организации веб–сессий, программный интерфейс расширений;
- является довольно полной заменой среды Microsoft ASP;
- может быть установлен практически на любой сервер;
- портирован под большое количество аппаратных платформ и операционных систем.

PHP также имеет ряд недостатков, среди которых:

- подходит только для создания web-приложений;
- слабые средства для работы с исключениями;
- объекты передаются по значению, а не по ссылке, как в большинстве других языков;
- зачастую проблемы с безопасностью[9].

1.3.3 Ruby

Ruby–интерпретируемый объектно-ориентированный язык программирования. Синтаксис основан на Perl, Smalltalk и Eiffel. Входит в десятку наиболее популярных языков программирования.

Обладает динамической типизацией и автоматическим управлением памятью. Ruby используется в веб-разработке в составе открытого веб-фреймворка Ruby on Rails (сокращённо RoR). Файлы программ, написанных на Ruby, имеют расширение .rb.

Преимущества языка Ruby:

- открытость;
- поддержка многих платформ;
- возможность внедрения в HTML;
- обладает высоким уровнем абстракции и предметным подходом в реализации алгоритмов;
- реализует концептуально чистую объектно-ориентированную парадигму;
- предоставляет продвинутые методы манипуляции строками и текстом;
- легко интегрирует в свои программы высокопроизводительные серверы баз данных (DB2, MySQL, Oracle и Sybase);
- хорошая масштабируемость;
- простой синтаксис;
- простой программный интерфейс для создания многопоточных приложений;
- имеет продвинутые средства для работы с массивами;
- возможности языка можно расширить при помощи библиотек, написанных на C или Ruby;
- зарезервированные слова могут являться идентификаторами;
- дополнительные возможности для обеспечения безопасности;
- встроенный отладчик.
- Недостатки языка Ruby:

- непростое обучение выше начального уровня;
- недостаток информационных ресурсов;
- меньшая производительность по сравнению со многими другими языками, применяемыми в веб-разработке;
- относительно медленно разрабатывается и развивается[9].

1.3.4 Python

Python– интерпретируемый язык для скриптов различного назначения (хотя существуют и трансляторы языка Python).

Целью Python, как и Ruby, является приближение синтаксиса реальной программы к псевдокоду, который описывает задачу, что позволяет уменьшить объём программы.

Особенности дизайна и синтаксиса этого языка облегчают программистам совместную работу над кодом.

Python является мультипарадигмальным языком программирования, то есть позволяет совмещать процедурный подход с объектно-ориентированным и функциональным.

Интерпретатор языка Python можно использовать как для запуска скриптов, так и в режиме интерактивной оболочки.

Преимущества языка Python:

- открытость;
- простота в изучении;
- особенности синтаксиса позволяют писать хорошо читаемый код;
- предоставляет средства быстрого прототипирования и динамической семантики;
- обширное сообщество;
- множество полезных библиотек и расширений языка можно легко использовать в своих проектах благодаря предельно унифицированному механизму импорта и программным интерфейсам;
- хорошо продуманы механизмы;
- всё является объектами в смысле объектно–ориентированного программирования, но собственно объектный подход применять необязательно.

Недостатки языка Python:

- неудачная поддержка многопоточности;
- довольно небольшое число качественных программных проектов по сравнению с другими универсальными языками программирования;
- отсутствует коммерческая поддержка средств;
- изначальная ограниченность средств для работы с базами данных;
- меньшая производительность Python по сравнению с основными Java VM [9].

1.3.5 JavaScript

JavaScript – объектно-ориентированный скриптовый язык программирования. Чаще всего использующийся используется для создания интерактивных страниц, хотя может выполняться и на стороне сервера (NodeJS).

Код на данном языке может быть как непосредственно прописан в HTML коде страницы внутри тега <script>, так и в отдельном файле.

Основные области применения JavaScript при создании интерактивных веб-страниц:

- динамическое создание содержимого страницы во время ее загрузки или после её полной загрузки;
- отображение диалоговых панелей и сообщений в статусной строке браузера;
- проверка достоверности данных в полях форм, заполняемых пользователем;
- динамическое изменение стилей и манипуляция объектной модели документа[10].

Для JavaScript написано множество фреймворков и библиотек. Самым широко используемым из них является JQuery.

JQuery позволяет существенно сократить код на языке JavaScript, и, тем самым ускорить разработку приложений.

Также существует дополнение JQueryUI, позволяющее программисту легко добавлять на страницу элементы пользовательского интерфейса, отсутствующие в стандарте HTML. Особенно стоит отметить элемент DatePicker – интерактивный календарь. К сожалению, на данный момент не все браузеры (в частности, Mozilla Firefox), поддерживают тип поля "дата", прописанный в стандарте HTML5. Как следствие, для реализации календаря на веб-странице, разработчику в большинстве случаев приходится использовать именно JQuery [11].

1.4 Системы управления базами данных

1.4.1 СУБД MySQL

MySQL – система управления базами данных (СУБД) с открытым исходным кодом. Очень часто применяется в сочетании с PHP[12].

MySQL является системой управления реляционными базами данных.

В MySQL применяется язык структурированных запросов SQL, наиболее распространенный стандартный язык, используемый для доступа к базам данных.

Для разработки веб-приложений предпочтительнее использовать именно СУБД MySQL, поскольку она является быстрой, надёжной и простой в использовании. [13]

MySQL является системой клиент-сервер, содержащей многопоточный SQL-сервер, который обеспечивает поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

Доступно также большое количество программного обеспечения для MySQL, в большей части – бесплатного.

Сервер MySQL постоянно работает на компьютере. Клиентские программы (например, скрипты, написанные на языке PHP) посылают серверу MySQL SQL-запросы при помощи сетевых средств, сервер их обрабатывает и запоминает результат. Клиент указывает, какую информацию он хочет получить от сервера баз данных, после чего сервер баз данных посылает ответ клиенту.

Результат запроса чаще всего передаётся не весь, поскольку объём данных может быть очень большим и на его пересылку уйдёт много времени

MySQL обладает трёхуровневой структурой: базы данных — таблицы — записи. Базы данных и таблицы MySQL физически представляются файлами с расширениями .frm, .myd, .myi. Логически таблица представляет собой совокупность записей, записи – совокупность полей разного типа. Имя базы данных MySQL уникально в пределах системы, а таблицы – в пределах базы данных, поля – в пределах таблицы. Один сервер MySQL может поддерживать сразу несколько баз данных, доступ к которым может разграничиваться логином и паролем. Зная логин и пароль, можно работать с конкретной базой данных. Обычно имя–идентификатор и пароль назначаются хостинг провайдерами, которые и обеспечивают поддержку MySQL для своих пользователей.

ДостоинстваMySQL:

- простота в работе и установке;
- поддержка большей части функционала SQL;
- безопасность;
- масштабируемость;
- скорость работы.

Недостатки MySQL:

- ограниченность функционала;
- иногда уступает другим СУБД по надежности;
- медленно разрабатывается [14].

1.4.2 СУБД PostgreSQL

PostgreSQL – свободно распространяемая СУБД. Максимально соответствует стандартам SQL.

От прочих СУБД PostgreSQL отличается поддержкой объектно-ориентированного и/или реляционного подхода к базам данных. Поддерживается параллельность благодаря реализации управления многовариантным параллелизмом. PostgreSQL поддерживаются хранимые процедуры, упрощающие использование часто выполняемых операций.

PostgreSQL не так популярна, как MySQL но существует большое число приложений облегчающих работу с PostgreSQL.

Достоинства PostgreSQL:

- открытость;
- обширное сообщество;
- большое число дополнений;

- поддержка объектно–ориентированных баз данных;
- высокая надёжность.
- Недостатки PostgreSQL
- проблемы с производительностью;
- меньшая популярность по сравнению с другими СУБД.
- сложно найти хостинг, поддерживающий PostgreSQL.

В качестве языка программирования выбран PHP, так как он наиболее пригоден для создания динамических веб–страниц.

В качестве СУБД выбрана MySQL, поскольку она является свободно распространяемой и чаще всего используется именно с PHP [14].

1.5 Среды разработки

1.5.1 NetBeans

NetBeans — свободная интегрированная среда разработки (сокращённо IDE), поддерживающая множество языков программирования, среди которых Java, Python, PHP, JavaScript, C, C++. Также поддерживается язык HTML и каскадные таблицы стилей CSS (в том числе полностью поддерживаются HTML 5 и CSS 3).

Существует специальная сборка NetBeans для разработки Web – приложений на PHP – NetBeans IDE Bubdle for PHP (рисунок 1.18).

Данная сборка включает в себя:

- подсветку синтаксиса и ошибок, автоматическое завершение кода;
- подсветку параметров и неиспользуемых локальных переменных;
- поддержку фреймворков Zend, Symthony1, Symthony2, Yii;
- отладчик xdebug.

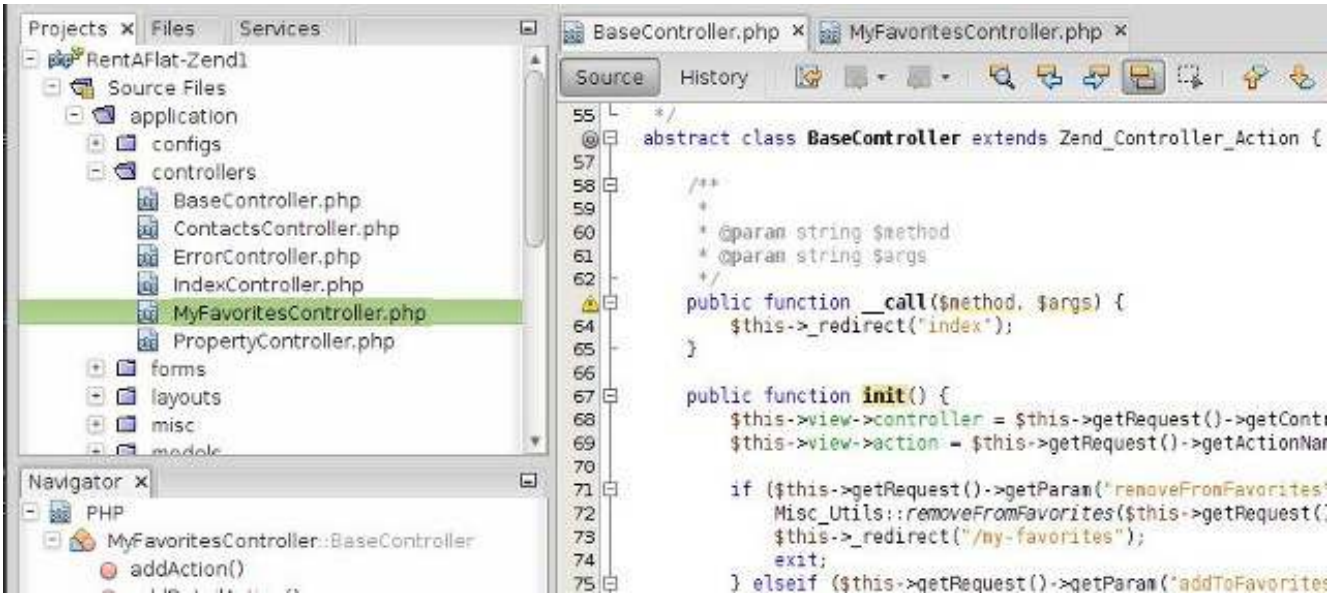

Рисунок 1.18 – NetBeans IDE Bubdle for PHP

1.5.2 PhpStorm

PhpStorm – коммерческая интегрированная среда разработки.

PhpStorm представляет собой интеллектуальный редактор для PHP, HTML, CSS и JavaScript с возможностями анализа кода "на лету" и предотвращения ошибок в коде. Полностью поддерживается HTML5 (рисунок 1.19).

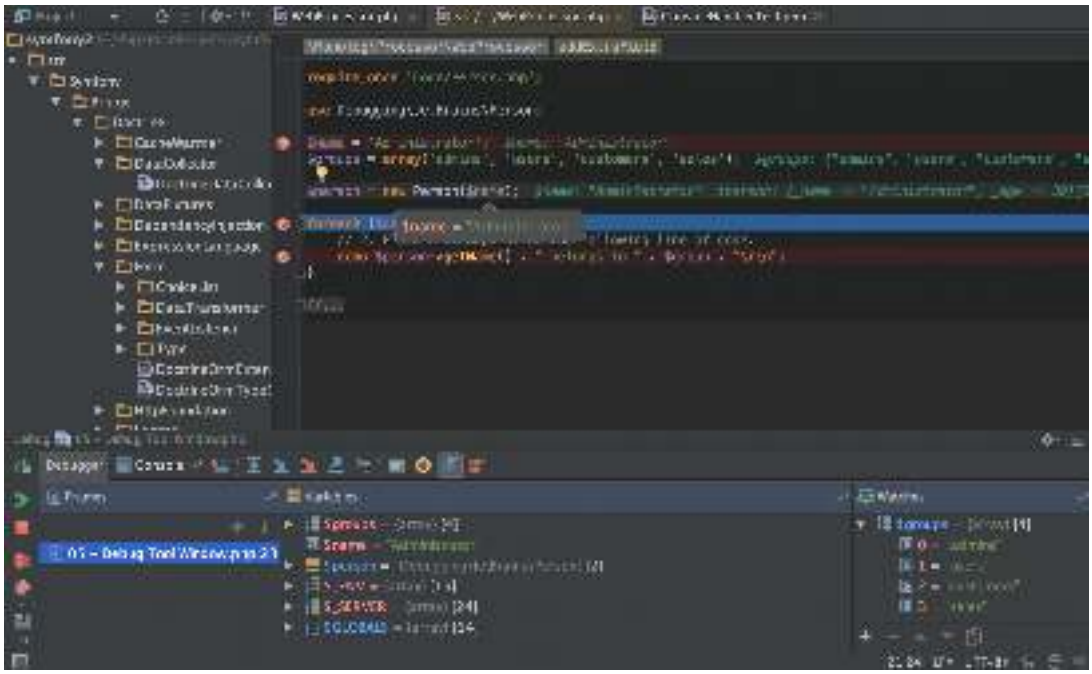

Рисунок 1.19 – PhpStorm

Поддерживаются спецификации PHP 5.3–5.6, 7.0 и 7.1. Также имеется полноценный SQL–редактор с возможностью редактирования полученных результатов запросов.

Функциональность среды разработки может быть расширена за счет установки плагинов.

Поддерживаются фреймворки Symfony2 и Yii.

Среда интегрирована с системами управления версиями GitHub, Subversion, Mercurial, Perforce, CVS, TFS.

Поддерживаются отладчики Xdebug и Zend Debugger.

1.5.3 Eclipse

Eclipse— свободная интегрированная среда разработки кроссплатформенных приложений. Первоначально среда была разработана для разработки приложений на языке Java, но в настоящее время существуют многочисленные расширения для поддержки множества языков, таких, как C ,C++, PHP, JavaScript, Perl, Python, Ruby и ряда других (рисунок 1.20).

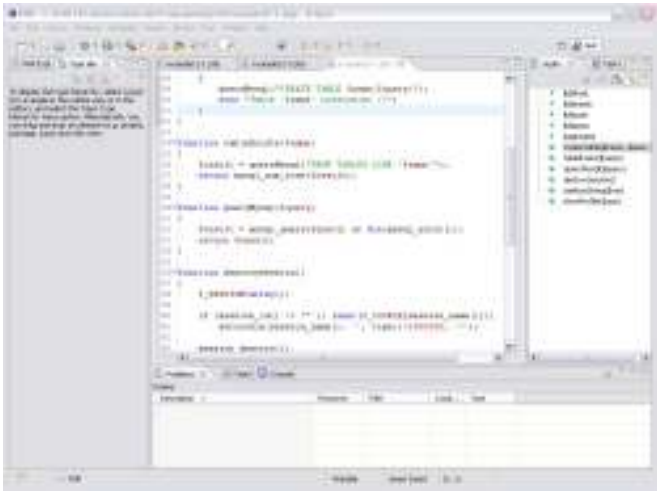

Рисунок 1.20 – Eclipse

Существует специальный пакет Eclipse для разработки Web - приложений на PHP – Eclipse for PHP Developers.

Данный пакет включает в себя Git клиент, Mylyn и редакторы для JavaScript, HTML, CSS и XML.

1.6 Постановка задачи

После анализа предметной области была уточнена цель работы: разработка модуля «Биржа труда для студентов» для сайта факультета Математики, механики и компьютерных технологий, с учетом дополнительных требований.

Целевой аудиторией являются выпускники, магистры, аспиранты и студенты последних курсов факультета Математики, механики и компьютерных технологий (далее – соискатели).

Требования к системе:

1. Требования к функциональным характеристикам

Система должна представлять совокупность методических и программных средств решения следующих задач:

- регистрация соискателей и работодателей;
- изменение личной информации соискателей и работодателей;
- добавление вакансий;
- добавление резюме;
- поиск вакансий;
- поиск резюме;
- публикация новостей о встречах, практиках, курсах и стажировках работодателей;
- обратная связь между соискателями и работодателями;

2. Требования к безопасности

Должна быть обеспечена безопасность и конфиденциальность личных данных пользователей.

3. Требования к информационной и программной совместимости

Браузер должен поддерживать стандарты HTML5 и CSS3, а также должна быть включена обработка JavaScript.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) выполнить обзор существующих сайтов для поиска работы;
2) провести анализ существующих технологий создания сайтов;
3) выбрать язык и платформу проектирования;
4) разработать архитектуру системы;
5) разработать базу данных;
6) разработать и реализовать алгоритмы системы;
7) выполнить проверку работы системы.

1.7 Выводы по разделу

Сайт FutureToday является наиболее подходящим для поиска работы студентом. Доступна информация о стажировках, работодатели могут общаться со студентами, сообщать о семинарах и прочих важных мероприятиях. Также вакансии могут быть предоставлены по специальности студента, если ВУЗ зарегистрируется на сайте.

Проблемой всех рассмотренных сайтов является поиск. На FutureToday и 74.ru поиск достаточно подробен, но не учитывает требований к соискателю, таких, как знание определённого языка или технологии. В модуле «Работа для студентов факультета ММиКН» частично решена эта проблема, но поиск там проводится только по навыкам. Также не учитываются такие важные критерии, как уровень владения тем или иным навыком.

Для создания модуля «Биржа труда для студентов» были выбраны технологии PHP, MySQL, HTML5 и среда разработки NetBeans.

2 РАЗРАБОТКА БАЗЫ ДАННЫХ

Процесс проектирования базы данных разбивается на 3 основных этапа.

1. Концептуальное проектирование – формулирование, анализ и выделение основных требований к данным.

2. Логическое проектирование – преобразование концептуальных требований в структуры данных.

3. Физическое проектирование – определение особенностей хранения данных, методов доступа и реализация этих особенностей на основе одной из существующих СУБД.

Перед тем, как начать проектирование базы данных, необходимо сформировать понятия о предметах, фактах и событиях, которыми будет оперировать создаваемая система[15].

Для того чтобы привести основные понятия предметной области к той или иной модели данных, необходимо заменить их информационными представлениями. Одним из наиболее удобных инструментов унифицированного представления данных, независимого от реализующего его программного обеспечения, является модель "сущность-связь" (Entity-Relationship Model).

2.1 Выделение сущностей предметной области

В качестве основных для системы «Биржа труда» в базе данных выделены следующие сущности: компания, вакансия, пользователь, резюме, новость, сообщение.

Компания должна обладать следующими свойствами:

- название компании;
- адрес компании;
- сайт компании;
- телефоны компании;
- краткая информация или примечания.

Пользователь должен обладать следующими свойствами:

- электронный адрес;
- пароль;
- фотография (аватар);
- фамилия, имя, отчество;
- дата регистрации;
- дата рождения.

Всего существует два типа пользователей: студент и представитель компании.

Пользователь должен обладать следующими свойствами:

- электронный адрес;
- пароль;
- фотография (аватар);

- фамилия, имя, отчество;
- дата регистрации;
- дата рождения.

Представитель компании обладает единственным свойством "должность". Представитель компании может размещать на сайте информацию о вакансиях. Свойства сущности «Вакансия»:

- название вакансии;
- занятость;
- личные качества;
- профессиональные навыки;
- обязанности;
- зарплата.

Студент может размещать на сайте своё резюме.
Резюме должно обладать следующими свойствами:

- направление обучения;
- персональная информация;
- навыки;
- желаемая должность;
- желаемая зарплата;
- желаемая занятость.

Представитель может размещать на сайте новости.
Новость должна обладать следующими свойствами:

- заголовок;
- краткий текст;
- полный текст;
- дата публикации;
- дата окончания актуальности.

Пользователя могут посылать друг другу сообщения.
Сообщение должно обладать следующими свойствами:

- заголовок;
- текст сообщения;
- дата и время отправки;
- информация о том, прочитано сообщение или нет.

2.2 ER-диаграмма сущностей

ER-диаграмма сущностей приведена на рисунке 2.1.

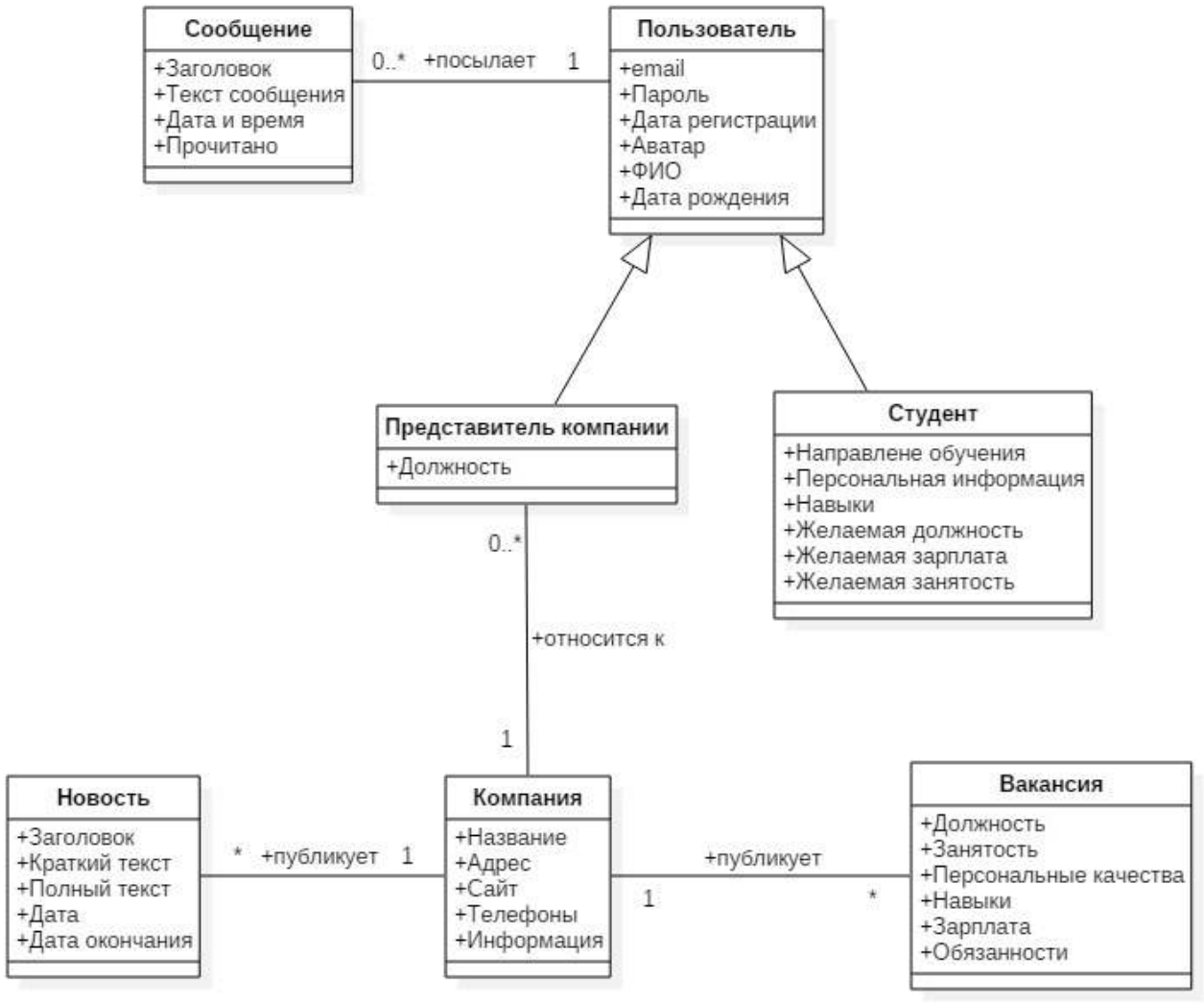

Рисунок 2.21 – ER-диаграмма сущностей

2.3 Разработка реляционной базы данных

Для преобразования ER-диаграммы к схеме базы данных, сущности и отношения между ними были преобразованы в таблицы.

Таблица 2.1 устанавливает соответствие идентификатор компании – характеристики компании.

Для хранения информации о направлении обучения студента были введены таблицы direction – направление и dir_type – тип направления (таблицы 2.2 и 2.3 соответственно).

Также была введена таблица role – роли для хранения информации о типе пользователя (таблица 2.5). Отношения между таблицами «пользователь» и

«представитель компании», а также «пользователь» и «студент» явно не указаны в базе данных, а будут реализованы в самом приложении.

Кроме того, введена дополнительная таблица work – «работа» дл учёта студентов, нашедших работу по вакансии.

Остальные сущности были преобразованы в таблицы реляционной базы данных без изменений с добавлением соответствующих идентификаторов (таблицы 2.1, 2.4, 2.6-2.12).

Таблица 2.1

Структура таблицы company

Поле	**Тип**	**Null**	**По умолчанию**
company_id	int(11)	Нет	
name	varchar(50)	Нет	
addr	varchar(50)	Нет	
site	varchar(50)	Нет	
phones	varchar(100)	Да	NULL
info	text	Да	NULL

Таблица 2.2

Структура таблицы dir_type

Поле	**Тип**	**Null**	**По умолчанию**
dir_type_id	int(11)	Нет	
type	varchar(50)	Нет	

Таблица 2.3

Структура таблицы direction

Поле	**Тип**	**Null**	**По умолчанию**
direction_id	int(11)	Нет	
dir_type_id	int(11)	Нет	
name	varchar(50)	Нет	

Таблица 2.4

Структура таблицы message

Поле	**Тип**	**Null**	**По умолчанию**
message_id	int(11)	Нет	
from_id	int(11)	Нет	
to_id	int(11)	Нет	
subject	varchar(50)	Нет	
text	text	Нет	
dateTime	datetime	Нет	
read	tinyint(1)	Нет	

Таблица 2.5

Структура таблицы role

Поле	**Тип**	**Null**	**По умолчанию**

role_id	int(11)	Нет	
roleName	varchar(50)	Нет	

Таблица 2.6

Структура таблицы news

Поле	**Тип**	**Null**	**По умолчанию**
news_id	int(11)	Нет	
company_id	int(11)	Нет	
subject	varchar(50)	Нет	
text	text	Нет	
datePost	date	Нет	
dateEnd	date	Нет	

Таблица 2.7

Структура таблицы represent

Поле	**Тип**	**Null**	**По умолчанию**
user_id	int(11)	Нет	
company_id	int(11)	Да	NULL
position	varchar(50)	Нет	

Таблица 2.1

Структура таблицы resume

Поле	**Тип**	**Null**	**По умолчанию**
user_id	int(11)	Нет	
direction_id	int(11)	Нет	
personal	text	Нет	
skills	text	Нет	
wantedVacancy	varchar(50)	Нет	
wantedSalary	varchar(50)	Нет	
wantedShelude	varchar(50)	Нет	

Таблица 2.9

Структура таблицы user

Поле	**Тип**	**Null**	**По умолчанию**
user_id	int(11)	Нет	
email	varchar(50)	Нет	
password	varchar(32)	Нет	
role_id	int(11)	Нет	
registerDate	date	Нет	
avatar	varchar(50)	Нет	
fio	varchar(100)	Нет	
birthDate	date	Нет	

Таблица 2.10

Структура таблицы vacancy

Поле	**Тип**	**Null**	**По умолчанию**
vacancy_id	int(11)	Нет	
company_id	int(11)	Нет	
vacancyName	varchar(50)	Нет	
shelude	varchar(50)	Нет	
personal	text	Нет	
skills	text	Нет	
duties	text	Нет	
salary	varchar(50)	Нет	

Таблица 2.2

Структура таблицы work

Поле	**Тип**	**Null**	**По умолчанию**
user_id	int(11)	Нет	
vacancy_id	int(11)	Нет	
date	date	Нет	

Схема разработанной базы данных "birga" представлена на рисунке 2.2.

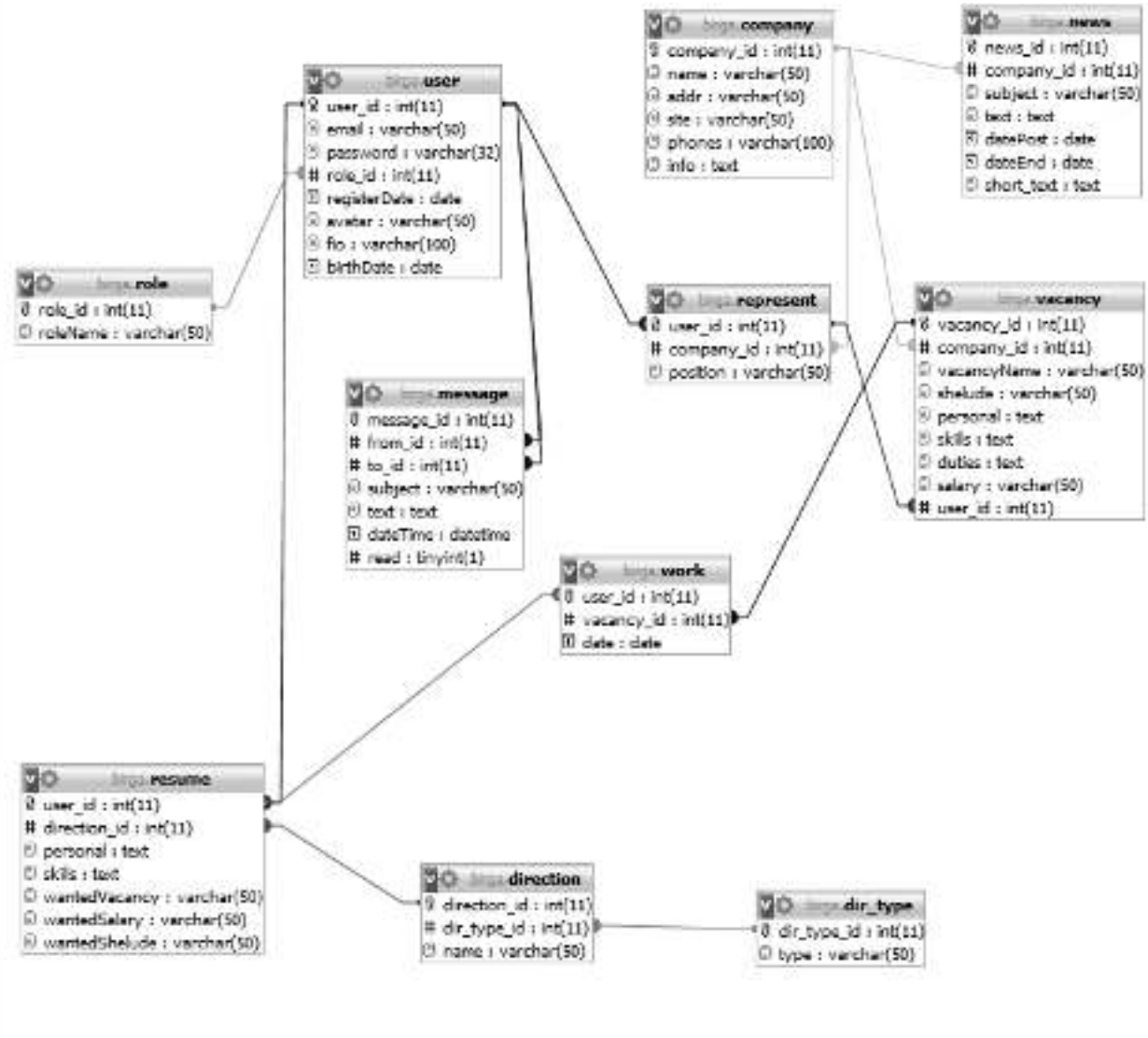

Рисунок 2.22 – Схема базы данных "birga"

2.4 Выводы по разделу

Выделены основные сущности предметной области, установлены отношения между ними. Сущности преобразованы в таблицы реляционной базы данных и приведена её схема.

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Разработка архитектуры приложения

В начале разработки приложения необходимо определить типы пользователей системы и их возможные действия[16].

Любой пользователь системы, в том числе и неавторизованный, может просматривать новости компаний, просматривать имеющиеся вакансии, а также фильтровать их список (то есть производить их поиск). Также неавторизованный пользователь может зарегистрироваться в системе.

Авторизованный пользователь (зарегистрированный пользователь) может осуществлять вход в систему, редактировать информацию на личной странице или страницах. Также у авторизованного пользователя существует возможность посылки личного сообщения другому пользователю.

Авторизованный пользователь типа «Соискатель» может публиковать своё резюме.

Авторизованный пользователь типа «Работодатель» может публиковать вакансии и новости, просматривать опубликованные резюме, а также фильтровать их список (то есть производить их поиск).

Диаграмма вариантов использования [17] представлена на рисунке 3.1.

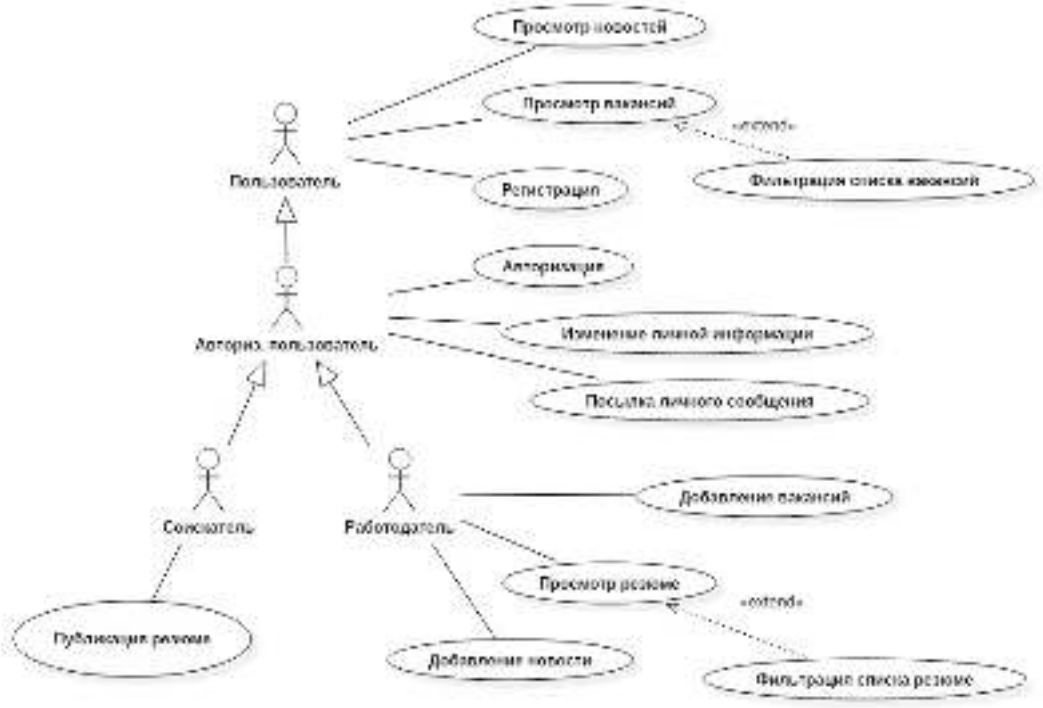

Рисунок 3.23 – Диаграмма вариантов использования

В качестве шаблона проектирования был взят шаблон Model-View-Controller (сокращённо MVC).

Целью применения данного шаблона является отделение обработки данных от конкретного пользовательского интерфейса, что позволяет в дальнейшем легко масштабировать и сопровождать приложение.

Компонент «Model» («Модель») отвечает за обработку данных, их предоставление и, в некоторых случаях, за проверку их корректности.

Компонент «View» («Представление») используется для отображения данных, полученных от контроллера из модели.

Компонент «Controller» («Контроллер») является связующим звеном между представлением и моделью. Отвечает за обработку действий пользователя, получение данных из модели и передачу данных в представление [18].

Схема шаблона проектирования MVC в общем виде представлена на рисунке 3.2.

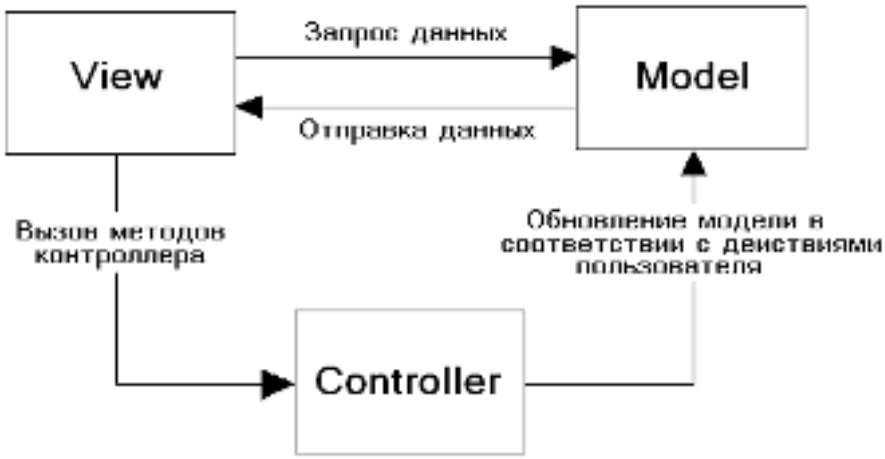

Рисунок 3.24 – Схема шаблона проектирования MVC

Реализация реализации шаблона MVC для данного приложения представлена на рисунке 3.3.

Рисунок 3.25 – Общая схема работы приложения

Модели представляют собой наборы процедур, выполняющих SQL запросы к базе данных.

Родительский класс «Модель» содержит поле, являющееся объектом вспомогательного класса «База данных». Данный класс предназначен для облегчения работы со стандартной библиотекой языка PHP для работы с СУБД MySQL – mysqlli.

Так как контроллер может обращаться к нескольким моделям, для улучшения производительности и экономии памяти подключение к базе данных должно создаваться только один раз. Для этого вспомогательный класс «База данных» был реализован с использованием паттерна проектирования «Одиночка». Он гарантирует, что существует только один экземпляр класса, и предоставляет к нему глобальную точку доступа [19].

Каждый контроллер соответствует одному пункту главного меню приложения.

Контроллер отвечает за пересылку данных, полученных от моделей, представлению, осуществляет проверку корректности введённых данных, обрабатывает действия пользователя на странице.

Один контроллер может отвечать за несколько представлений, например, страницу со списком компаний и страницу с информацией о конкретной компании.

Контроллер может не иметь представлений и не обращаться ни к одной модели, как, например, контроллер «Выход».

Диаграмма классов контроллеров представлена на рисунке 3.4, диаграмма классов моделей представлена на рисунке 3.5.

Рисунок 3.26 – Диаграмма классов контроллеров

Рисунок 3.27 – Диаграмма классов моделей

Модуль «Роутер» предназначен для определения того, какой контроллер и какое действия контроллера необходимо вызвать.

URL запрос, приходящий от пользователя в результате его ручного ввода в адресную строку браузера или при переходе по ссылке, имеет следующий вид:

[адрес сайта]/[имя контроллера]/[имя действия]?[GET запрос]

Для роутера необходимы только имя контроллера и имя действия контроллера. Если они указаны и существуют, то соответствующее действие контроллера выполняется. Если не указаны имя контроллера не указано, то вызывается контроллер по умолчанию, а именно – контроллер главной страницы. Если же контроллер либо действие не существует, то выдаётся сообщение об отсутствии страницы.

Схема алгоритма модуля «Роутер» представлена на рисунке 3.6

Модуль «Представление» представляет собой программный код, формирующий страницу на языке разметки HTML и, в некоторых случаях, код на языке Javascript, на основе данных, полученных через действие контроллера от модели или моделей.

Схема алгоритма модуля «Представление» представлена на рисунке 3.7

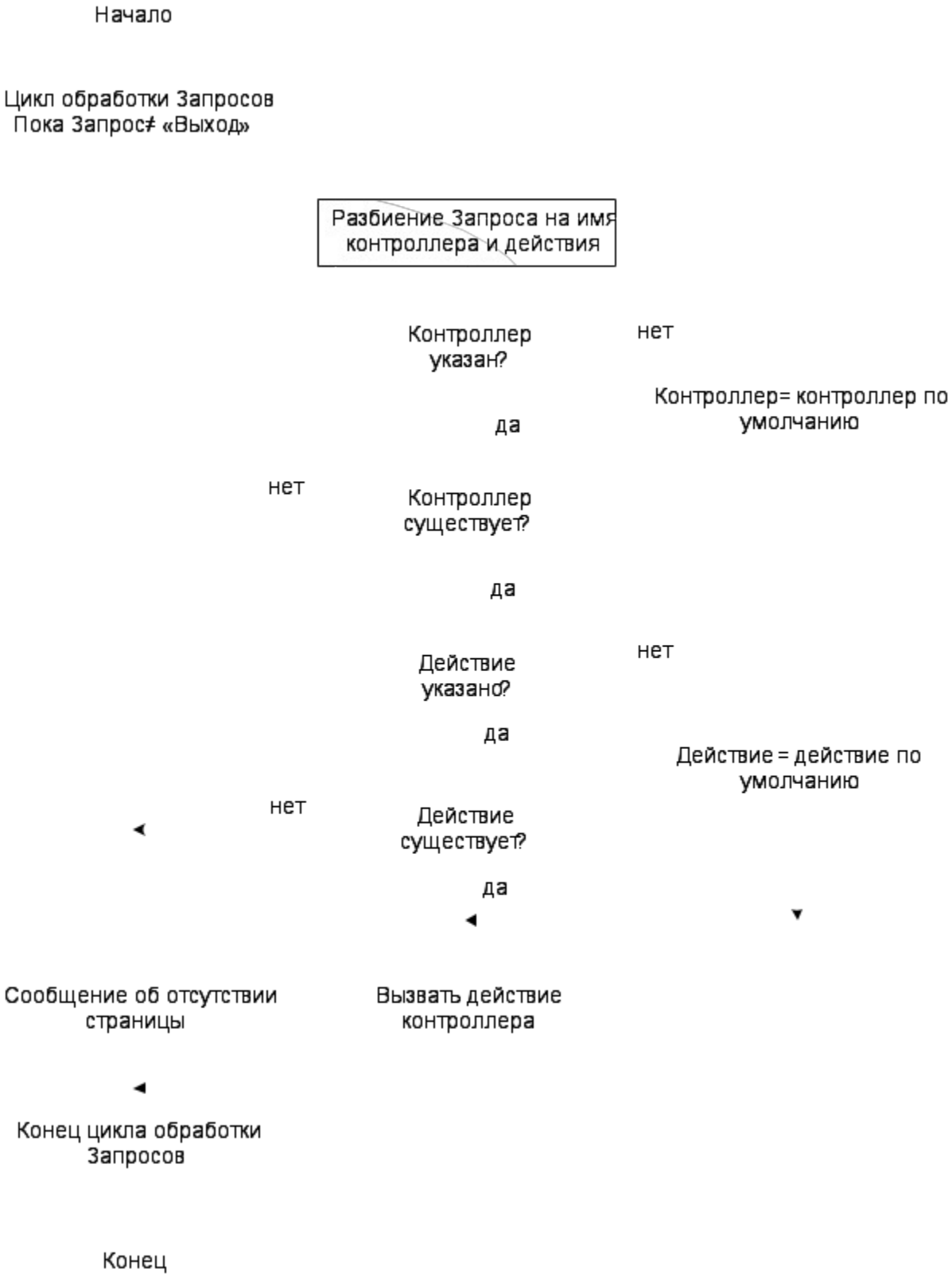

Рисунок 3.28 – Алгоритм модуля «Роутер»

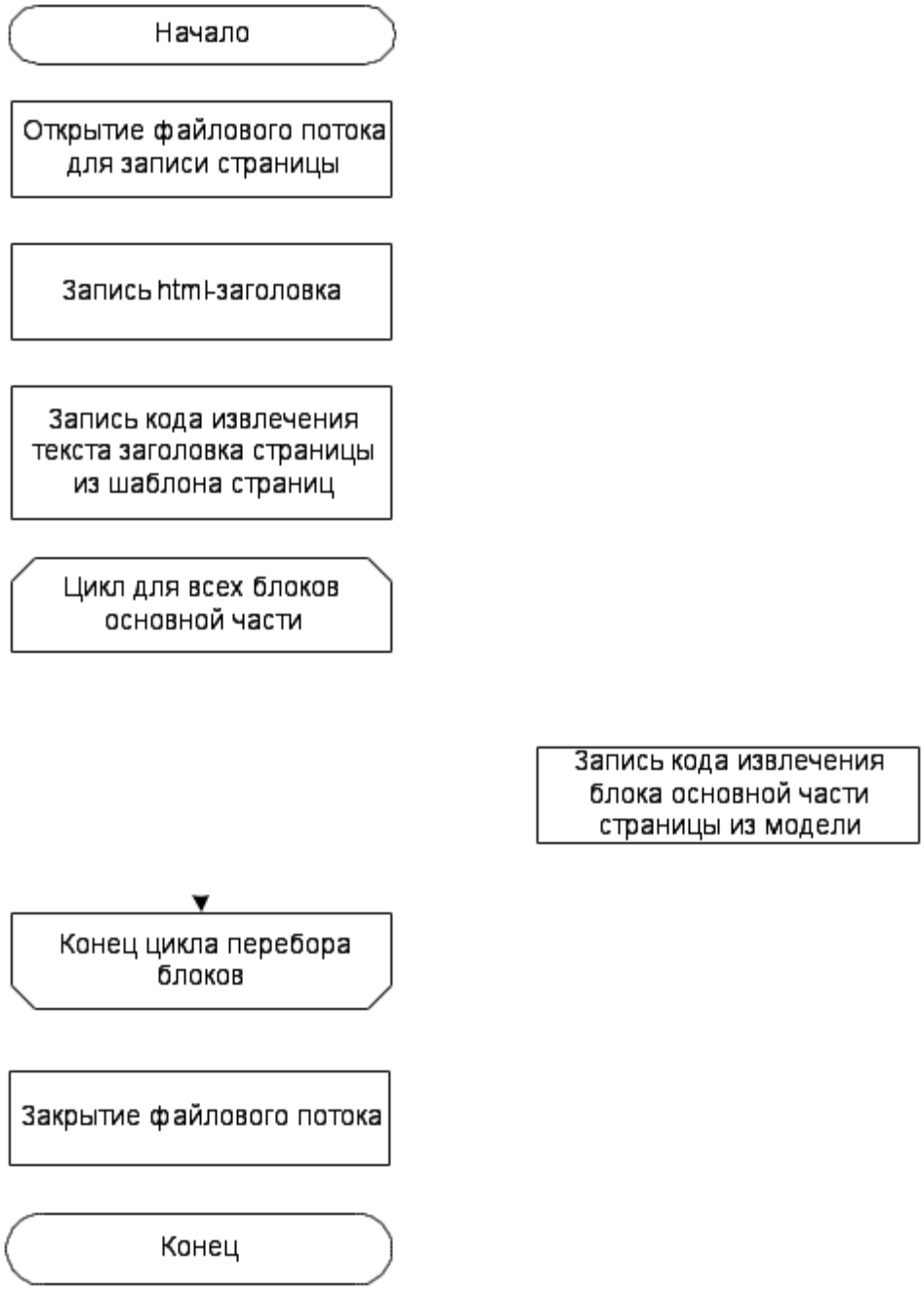

Рисунок 3.29 – Алгоритм модуля «Представление»

3.2 Разработка интерфейса

Так как разрабатываемая система будет расположена на сайте факультета Математики, механики и компьютерных технологий в центре страницы, в самой системе не должно быть заголовка страницы и нижней части страницы.

В рабочей области системы слева расположено главное меню, разделённое визуально на две части. В верхней её части расположены пункты, относящиеся непосредственно к пользователю. Для неавторизованного пользователя это пункты входа и регистрации, для авторизованного – пункт для перехода на личную страницу (личных страниц может быть несколько, например, для работодателя это непосредственно его личная страница и страница его компании) и пункт выхода из системы. Также в верхней части меню отображается аватар пользователя[20].

В нижней части меню располагаются пункты перехода на остальные страницы системы, такие, как главная страница, страница с новостями компаний, страница со списком компаний, страницы поиска резюме и вакансий.

Справа в рабочей области системы отображается содержимое страниц, на которые пользователь переходит, выбрав соответствующий пункт меню.

Общий вид всех страниц для неавторизованного пользователя показан на рисунке 3.8, для авторизованного – на рисунке 3.9.

Разного рода списки, такие, как списки новостей или списки компаний, представлены в виде последовательности блоков, содержащих краткую информацию и ссылки для перехода на страницу с более подробной информацией, а также, быть может, на другие страницы, например, на страницу автора.

Вид страницы со списком информационных блоков представлен на рисунке 3.10.

На страницах со списками, в которых предусмотрен поиск информации, сверху расположена форма поиска. Содержимое списка меняется в зависимости от параметров поиска, причем введённые пользователем параметры должны сохраняться при нажатии кнопки поиска

На некоторых страницах со списком информационных блоков, таких, как резюме или вакансий, предусмотрена фильтрация списка, или поиск по конкретным параметрам.

Форма поиска, содержащая поля параметров фильтрации и кнопку поиска, находится в верхней части блока содержания страницы.

Поля параметров фильтрации могут быть как текстовыми, так и различными переключателями, счётчиками, выпадающими списками.

После нажатия на кнопку поиска, список информационных блоков фильтруется в соответствии с введёнными параметрами. Если блоки, соответствующие данным параметрам, не были найдены, то вместо результатов поиска выводится соответствующее сообщение.

Вид страницы с поиском по списку информационных блоков представлен на рисунке 3.11.

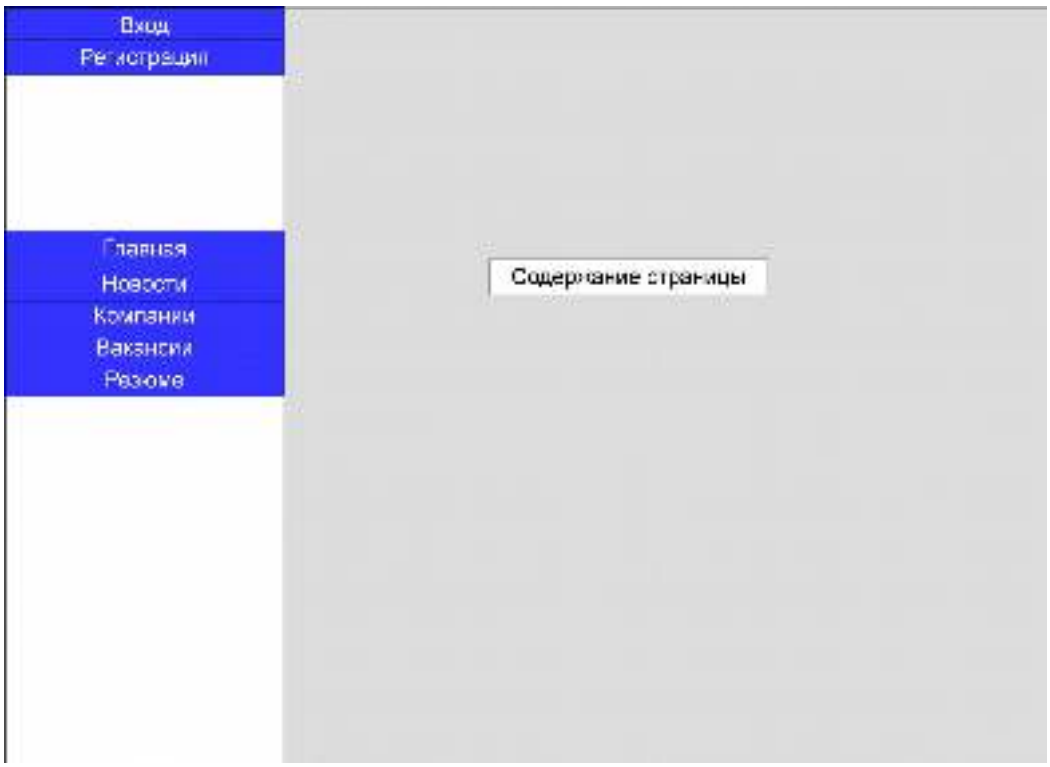

Рисунок 3.30 – Общий вид страниц для
неавторизованного пользователя

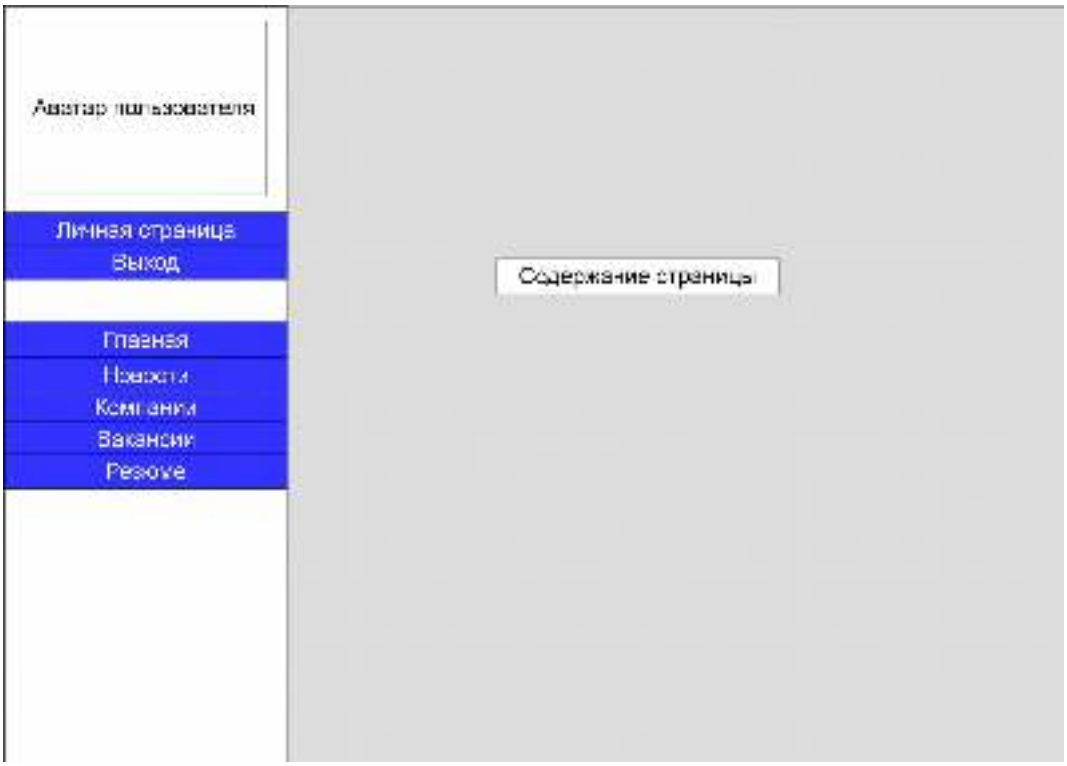

Рисунок 3.31 – Общий вид страниц
для авторизованного пользователя

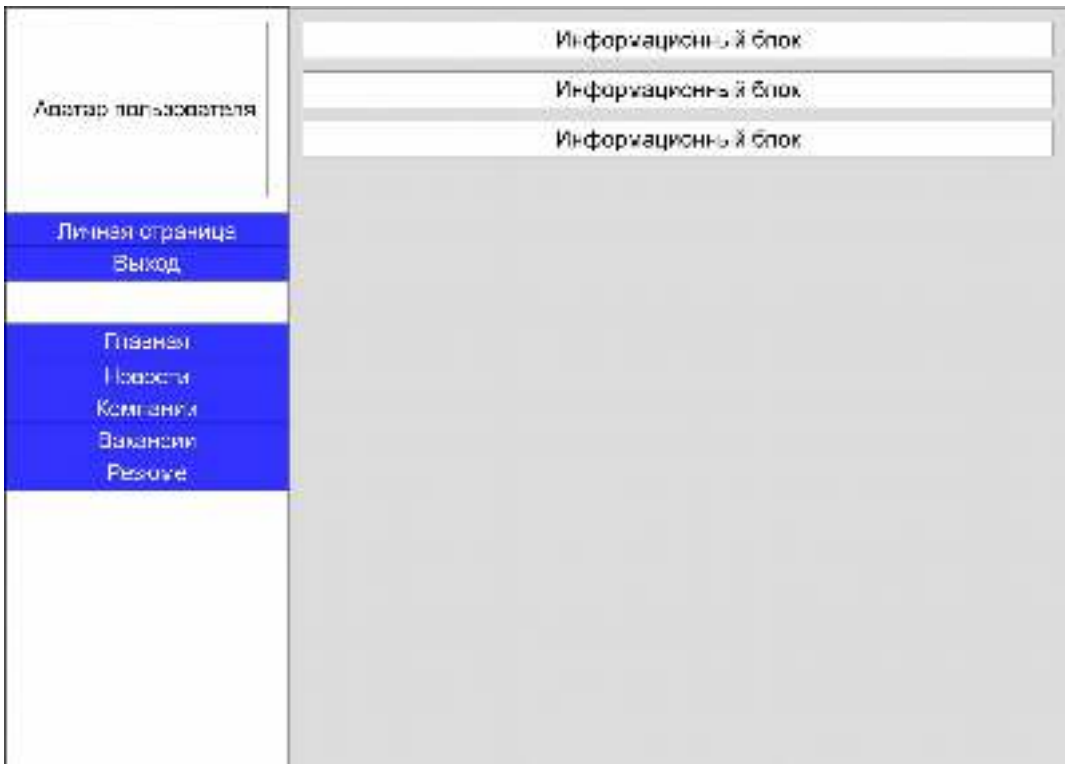

Рисунок 3.32 – Страница со списком информационных блоков

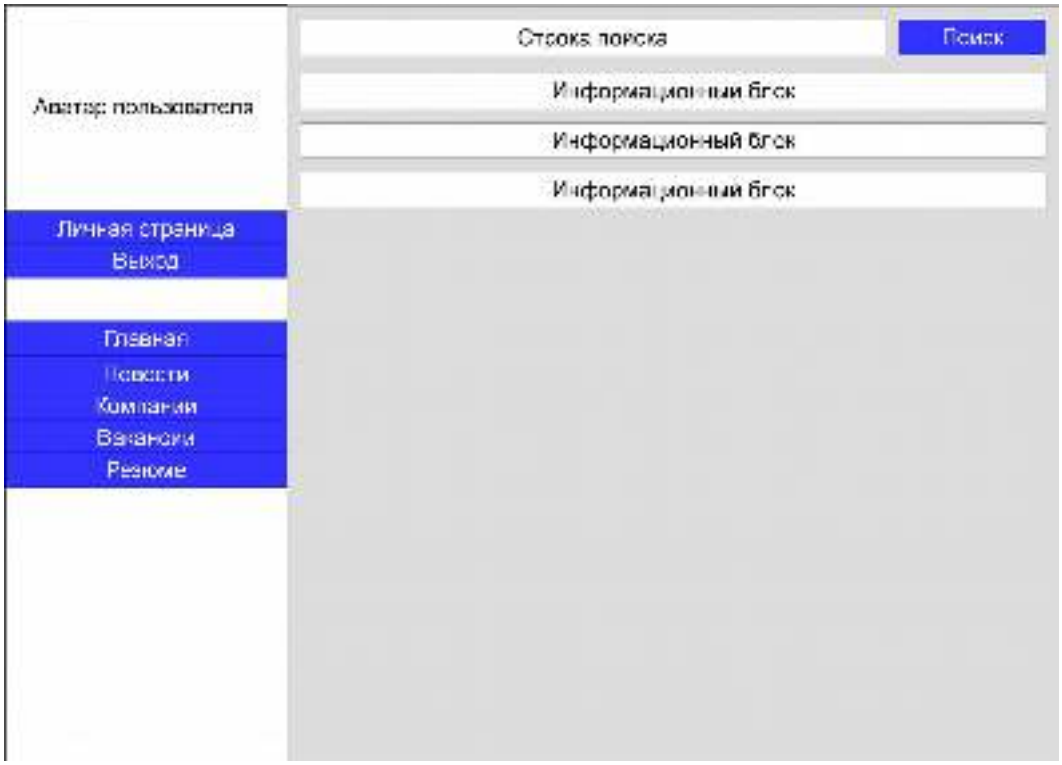

Рисунок 3.33 – Страница с поиском по списку информационных блоков

3.3 Выводы по разделу

В данном разделе разработана архитектура системы. Представлены диаграммы вариантов использования, диаграммы классов и схемы алгоритмов работы модулей. Кроме того, разработан интерфейс пользователя.

Диаграмма вариантов использования включает в себя все возможные действия пользователей системы. Диаграммы классов описывают структурные компоненты системы.

Диаграммы описаны с помощью языка UML. При разработке системы использован шаблон проектирования Model-View-Controller (MVC).

4 ПРОВЕРКА РАБОТЫ ПРИЛОЖЕНИЯ

4.1 Главная страница

На главной странице справа отображаются счётчики резюме и вакансий, опубликованных в системе на данный момент. Слева расположено главное меню, в нижней части которого располагаются ссылки на основные страницы, а именно: главную страницу, страницы новостей, компаний, резюме и вакансий. Данные ссылки доступны всем типам пользователей системы.

Для неавторизованного пользователя в меню сверху располагаются пункты входа в систему и регистрации (рисунок 4.1).

Для регистрации в системе необходимо заполнить форму, располагающуюся по ссылке «Регистрация». На данной форме, помимо полей для ввода персональной информации и пароля, присутствует переключатель типа пользователя. Тип по умолчанию – студент; после выбора типа «Работодатель» появляется поля для должности и список компаний, уже зарегистрированных на сайте. Если же нужной компании в списке нет, то необходимо выбрать пункт списка «Компании нет в списке». При регистрации необходимо ввести подтверждение пароля.

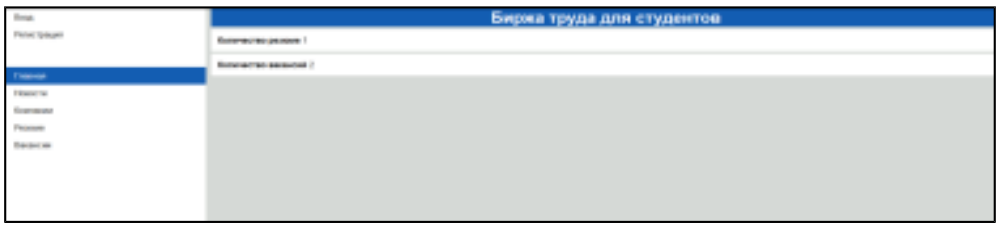

Рисунок 4.34 – Главная страница – пользователь не авторизован

Зарегистрированный пользователь может войти в систему. Для этого необходимо выбрать пункт меню «Вход» и ввести email и пароль в соответствующие поля появившейся формы.

После входа в систему в верхней части меню появляются аватар пользователя, пункты выхода из системы и перехода на личную страницу «Моя страница». Для типа пользователя «Представитель компании» дополнительно появляется пункт перехода на личную страницу компании «Моя компания» (рисунки 4.2 и 4.3).

После входа в систему пользователь остаётся авторизованным, пока не закроет окно браузера, либо пока не пройдёт 15 минут.

Для выхода из системы необходимо выбрать пункт меню «Выход».

После выхода из системы пользователь становится неавторизованным и автоматически перенаправляется на главную страницу.

Пункт меню, соответствующий текущей странице, выделяется цветом.

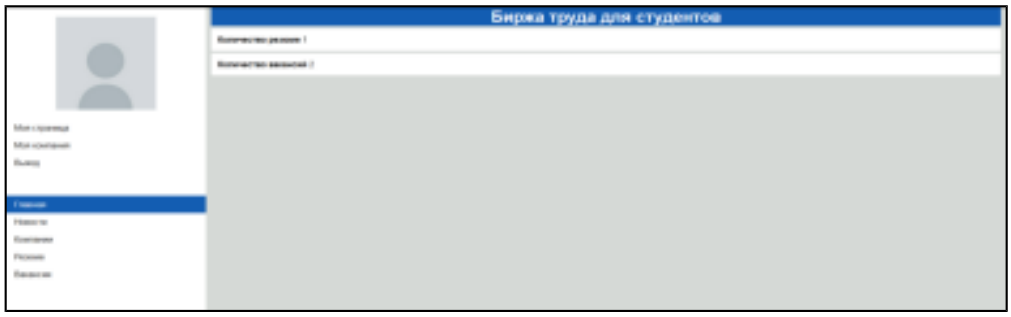

Рисунок 4.35 – Главная страница – пользователь представитель компании

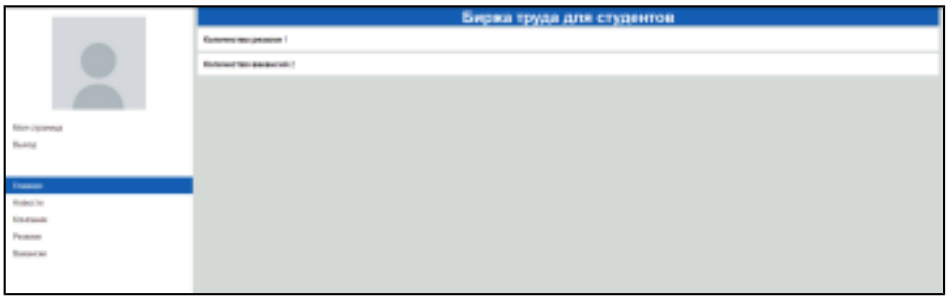

Рисунок 4.36 – Главная страница – пользователь студент

4.2 Личные страницы пользователей

При нажатии на пункт меню «Моя страница» пользователь переходит на страницу, содержащую его личную информацию. Для студента также отображается его резюме (рисунок 4.4). Если резюме ещё не было добавлено, то вместо него отображается кнопка «Добавить». После нажатия на неё, пользователь переходит на страницу, содержащую форму резюме.

Пункты резюме «О себе» и «Навыки» необходимо заполнять в формате «пункт-пункт-пункт…».

Также на личной странице в блоке персональной информации присутствует кнопка «Изменить информацию», при нажатии на которую пользователь переходит на страницу с соответствующей формой для редактирования информации.

Если представитель компании при регистрации не выбрал компанию, то при выборе пункта «Моя компания» он перейдёт на страницу с формой добавления новой компании.

Страница «Моя компания» содержит информацию о компании, новости, опубликованные представителями компании, а также вакансии, опубликованные данным представителем.

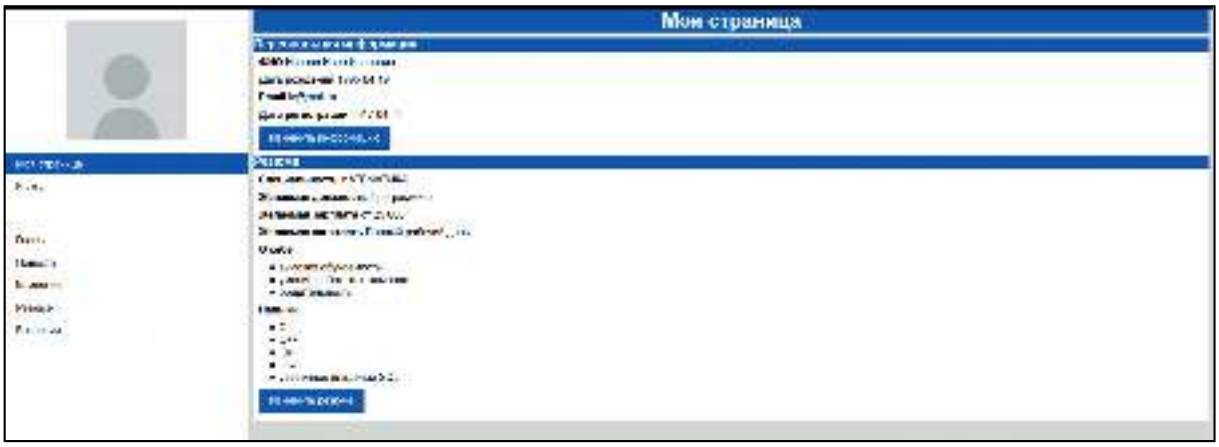

Рисунок 4.37 – Личная страница пользователя «Студент»

Списки новостей и вакансий изначально находятся в свёрнутом состоянии для экономии места. Их можно развернуть, нажав на ссылку «показать» (рисунок 4.5).

Рисунок 4.38 – Личная страница компании

На странице компании также присутствуют кнопки перехода на страницы добавления вакансий и новостей. Их содержание можно изменить, перейдя по соответствующей ссылке в блоке вакансии или резюме. Также можно перейти на их страницы, нажав на заголовок.

Стоит отметить, что изменять и добавлять информацию пользователь может только из личной страницы или страницы своей компании.

4.3 Страница новостей

На странице новостей отображается список блоков новостей. В блоке отображаются заголовок новости, краткий текст новости, ссылку на компанию, разместившую новость и дату размещения новости (рисунок 4.6).

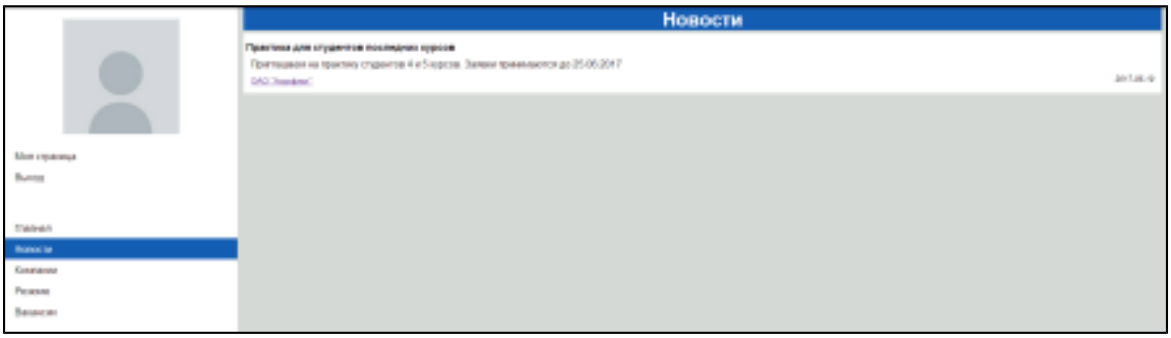

Рисунок 4.39 – Страница со списком всех новостей

Внизу страницы располагается ссылка на архив новостей. Новости, утратившие актуальность, попадают в архив автоматически. Архив представляет собой страницу, аналогичную странице со списком новостей.

При щелчке по заголовку новости пользователь может перейти на страницу, содержащую полный текст новости (рисунок 4.7).

Информация на данной странице аналогична блоку из списка, единственное отличие в том, что краткий текст новости заменяется на полный текст.

Рисунок 4.40 – Страница конкретной новости

4.4 Страница компаний

На странице компаний отображается список компаний, зарегистрированных в системе. (рисунок 4.8).

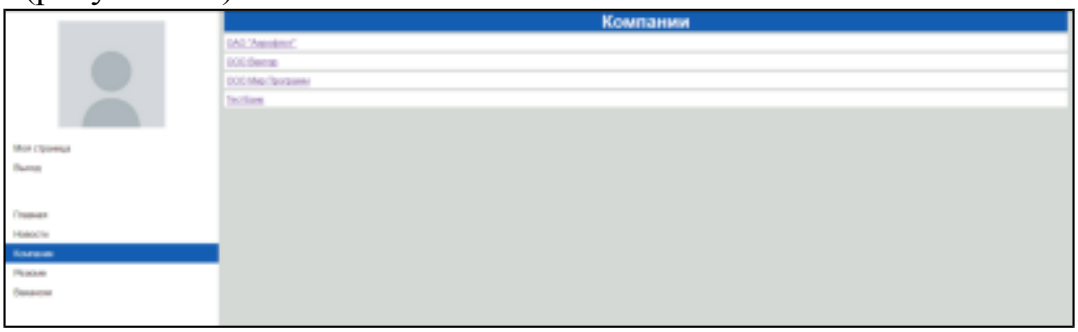

Рисунок 4.41 – Страница со списком компаний

При нажатии на ссылку, содержащую название компании, пользователь переходит на страницу компании.

Страница компании, по сути, представляет собой личную страницу компании без возможности редактирования. Также, в отличие от неё, она содержит информацию о зарегистрированных представителях данной компании, и отображаются все вакансии, а не только вакансии, добавленные конкретным представителем (рисунок 4.9).

Рисунок 4.42 – Страница конкретной компании

4.5 Страницы вакансий и резюме

На страницах вакансий и резюме сверху расположена форма поиска по названию должности. При переходе на данные страницы отображается полный список опубликованных вакансий или резюме. При нажатии кнопки «Поиск» отображается список вакансий или резюме, чья графа «Должность» соответствует строке, введённой в поле поиска (рисунки 4.10 и 4.11).

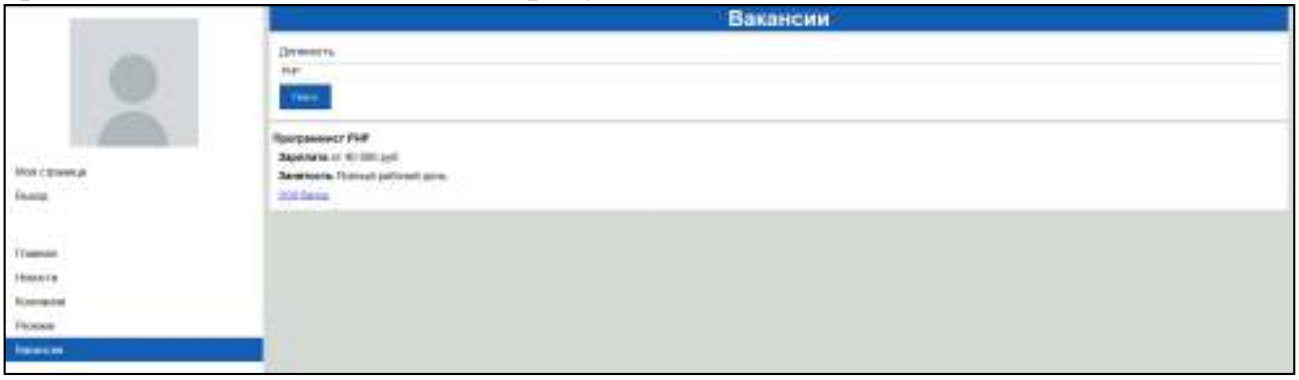

Рисунок 4.43 – Страница поиска вакансий

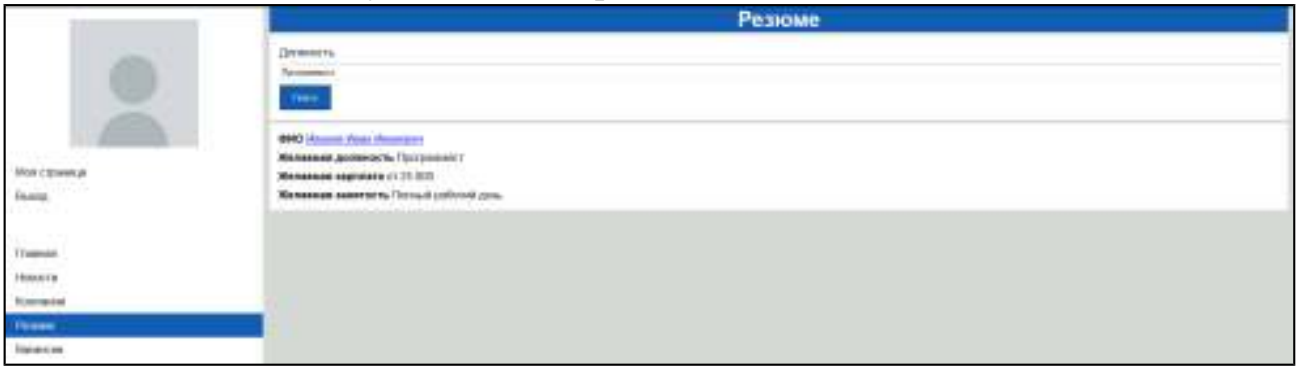

Рисунок 4.44 – Страница поиска резюме

Если ничего не было найдено, то вместо результатов поиска выводится соответствующее сообщение.

При щелчке по должности в блоке вакансии или ФИО студента в блоке резюме пользователь перейдёт на страницы с более полной информацией.

Помимо собственно резюме, на странице студента указаны email и дата рождения(рисунок 4.12).

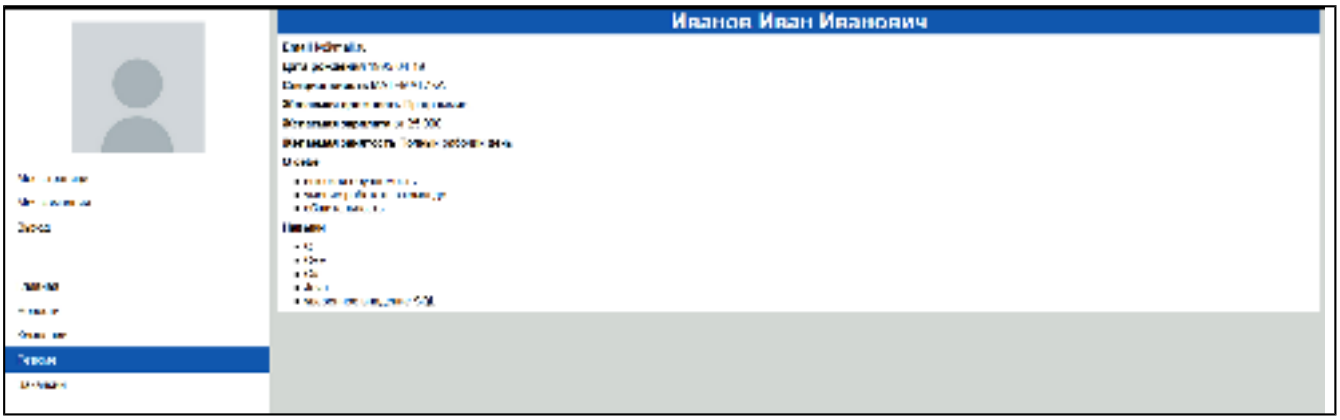

Рисунок 4.45 – Страница полного резюме

Просмотр страницы вакансии доступен всем пользователям, даже не авторизовавшимся. Однако, просмотр страницы с полным резюме доступен только представителям компании, вошедшим в систему (рисунок 4.13).

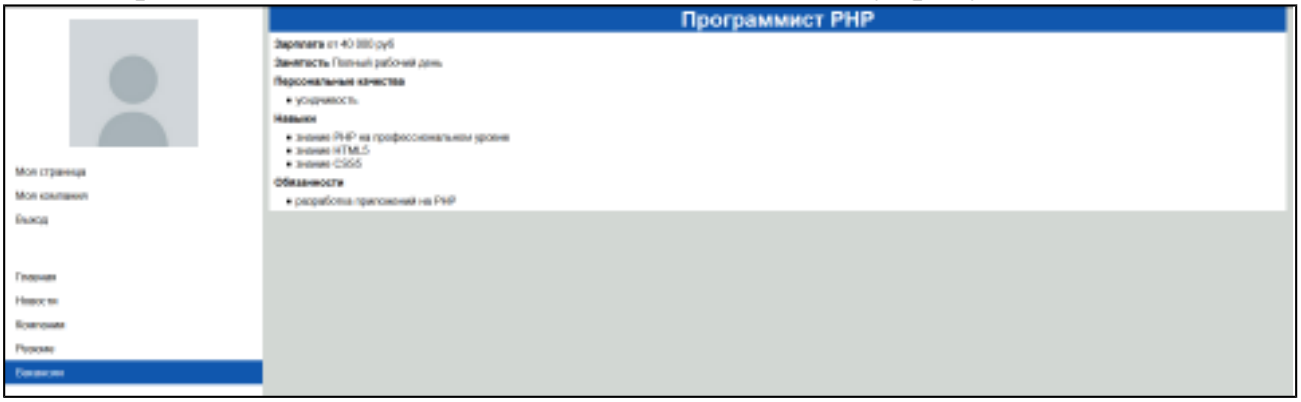

Рисунок 4.46 – Страница вакансии

4.6 Внедрение модуля

Модуль «Биржа труда для студентов» будет внедрён на сайт факультета Математики, механики и компьютерных технологий с помощью технологии плавающих фреймов[21]. Сам модуль физически располагается на сервере кафедры Прикладной математики и программирования.

Вид страницы сайта после внедрения приведён на рисунке 4.14.

Рисунок 4.47 – Вид страницы

с внедрённым модулем «Биржа труда для студентов»

4.7 Выводы по разделу

Проведена проверка функций системы с точки зрения разных типов пользователей.

Приведён вид страницы сайта факультета Математики, механики и компьютерных технологий после внедрения системы.

В первой версии отсутствует система личных сообщений между студентами и представителями компаний, а также учёт принятия студента на работу по вакансии. Данные функции будут реализованы при дальнейшей разработке данной системы.

ЗАКЛЮЧЕНИЕ

Работа посвящена разработке модуля «Биржа труда для студентов» сайта факультета Математики, механики и компьютерных наук.

Данный модуль позволяет облегчить процесс поиска работы студентам последних курсов. Пользователи данного модуля могут просматривать информацию о зарегистрированных компаниях, новостях о различных мероприятиях, проводимых ими и их текущих вакансиях. Также пользователи могут изменять опубликованную ими информацию на личной странице.

Пользователь типа «Студент» может размещать своё резюме. Пользователь типа «Представитель компании» может просматривать опубликованные резюме, публиковать вакансии и новости.

В результате работы:

- спроектирована база данных;
- разработан алгоритм работы системы;
- разработан графический интерфейс;
- выполнена программная реализация.

В дальнейшем планируется реализовать следующие функции системы:

- администрирование;
- личные сообщения между представителями компаний и студентами;
- более подробный поиск вакансий и резюме.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Карьера студентов факультета ММиКТ [Электронный ресурс]. URL: http://math.susu.ru/index.php/studentam/karera–testovyj–rezhim/ (дата обращения 21.03.2017).
2. FutureToday [Электронный ресурс]. URL: http://fut.ru/ (дата обращения 21.03.2017).
3. Зарплата.ру [Электронный ресурс]. URL: http://chelyabinsk. zarplata.ru /?gm (дата обращения 17.03.2017).
4. Статические или динамические сайты: что выбрать [Электронный ресурс]. URL: http://webstudio2u.net/ru/design-web/391-static-or-dynamic.html (дата обращения 15.03.2017).
5. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Б. Хоган.– СПб.: Питер, 2014. – 320 с.
6. Панфилов, К. Создание веб–сайта от замысла до реализации./К.Панфилов. – М.: ДМК Пресс, 2009. – 440 с.
7. Одиночкина, С.В. Web–программирование PHP. /.С.В.Одиночкина. – СПб. : НИУ ИТМО, 2012 – 79 с.
8. PHP: Hypertext Preprocessor [Электронный ресурс]. URL: http://php.net// (дата обращения 22.03.2017).
9. PHP, Ruby, Python – краткая характеристика трёх языков программирования [Электронный ресурс]. URL: http://www.internet–technologies.ru/articles /article_1991.html/ (дата обращения 27.03.2017).
10. Зудилова, Т.В. Web-программирование JavaScript. / Т.В. Зудилова, М.Л. Буркова. – СПб. : НИУ ИТМО, 2012. – 68 с.
11. Перепелица, Ф.А. Разработка интерактивных сайтов с использованием jQuery / Ф.А.Перепелица. – СПб. : НИУ ИТМО, 2015. – 142 с.
12. Ульман, Л. MySQ /Л.Ульман. – М. : ДМК Пресс, 2008. – 352 с.
13. MySQL [Электронный ресурс]. URL: https://www.mysql.com/ (дата обращения 21.03.2017).
14. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных [Электронный ресурс]. URL: http://devacademy.ru/posts/sqlite–vs–mysql–vs–postgresql/ (дата обращения 19.03.2017).
15. Тарасов, С.В. СУБД для программиста. Базы данных изнутри./С.В.Тарасов. – М. : СОЛОН-Пресс, 2015. – 320 с.
16. Буч, Г. Язык UML. Руководство пользователя. / Г. Буч, Д. Рамбо, И. Якобсон. – М. : ДМК Пресс, 2008. – 496 с.
17. Иванов, Д. Моделирование на UML / Д. Иванов, Ф. Новиков. – СПб. : НИУ ИТМО, 2010. – 200 с.
18. Реализация MVC паттерна на примере создания сайта-визитки на PHP [Электронный ресурс]. URL: https://habrahabr.ru/post/150267/. (дата обращения 24.03.2017).

19. Гамма, Э. Приемы объектно ориентированного проектирования. Паттерны проектирования. / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес.– М. : ДМК Пресс, 2007. – 368 с.
20. Мандел, Т. Разработка пользовательского интерфейса./ Т.Мандел. – М. : ДМК Пресс, 2007. – 418 с.
21. W3Schools Online Web Tutorials [Электронный ресурс]/ URL: https://www.w3schools.com/ (дата обращения 21.03.2017).

ПРИЛОЖЕНИЕ 1

ТЕКСТ ПРОГРАММЫ

Исходные тексты модулей системы

index.php – точка входа

```
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL & ~E_NOTICE);
require_once 'application/bootstrap.php';
```

.htaccess – файл настроек сервера

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule .* index.php [L]//перенаправление запросов на index.php
```

bootstrap.php – файл с подключениями файлов ядра

```
<?php
// подключение файлов ядра
require_once 'core/model.php';
require_once 'core/view.php';
require_once 'core/controller.php';
require_once 'core/route.php';
require_once 'core/database.php';
require_once 'core/ui.php';
Route::start(); // запуск маршрутизатора
```

route.php – файл класса роутера

```
<?php
class Route
{
   //запуск роутера
   static function start()
   {
   // контроллер и действие по умолчанию
   $controller_name = 'Main';
   $action_name = 'index';
   $url = $_SERVER['REQUEST_URI'];
   $url = strtok($url, '?');
   $routes = explode('/', $url);
   // получаем имя контроллера
   if ( !empty($routes[1]) )
   {
   $controller_name = $routes[1];
   }
   // получаем имя экшена
   if ( !empty($routes[2]) )
   {
   $action_name = $routes[2];
   }
   // добавляем префиксы
   $controller_name = 'Controller_'.$controller_name;
   $action_name = 'action_'.$action_name;
   // подцепляем файл с классом контроллера
   $controller_file = strtolower($controller_name).'.php';
   $controller_path = "application/controllers/".$controller_file;
   if(file_exists($controller_path))
   {
    include $controller_path;
   }
```

```
      else
      {
       Route::ErrorPage404();
      }
      // создаем контроллер
      $controller = new $controller_name;
      $action = $action_name;

      if(method_exists($controller, $action))
      {
      // вызываем действие контроллера
      $controller->$action();
      }
      else
      {
      Route::ErrorPage404();
      }
      }
      //обработка ошибки 404
      function ErrorPage404()
      {
      $controller_path = "application/controllers/controller_404.php";
      include $controller_path;
      $controller = new Controller_404;
      $controller->action_index();
      }
   }
```

database.php – файл класса работы с базой данных

```
   <?php
   class Database{
      private $conn;//соединение с БД
      private static $db = null;//ссылка на объект БД
      //получить ссылку на объект БД
      public static function getDB()
      {
      include 'application/config/db_config.php';
      if (self::$db == null) self::$db = new Database($db_host, $db_user,
$db_password, $db_name);
      return self::$db;
      }

      private function __construct($db_host, $db_user, $db_password, $db_name)
      {
      @$this->conn = $mysqli = new mysqli($db_host, $db_user, $db_password,
$db_name);
      @$this->conn->set_charset("utf8");
      if(mysqli_connect_errno($this->conn)) {
      throw new Exception("Не удалось подключиться к базе данных");
      }
      }
      //удаление спец символов
      public function escape($s)
      {
      return mysqli_escape_string($this->conn, $s);
      }
      public function ins_id()
      {
      return mysqli_insert_id($this->conn);
      }
      //выполнение запроса
      public function query($q){
```

```
        $result = $this->conn->query($q);

        if (mysqli_error($this->conn)){
         throw new Exception(mysqli_error($this->conn));
        }

        if (is_bool($result)){
        return $result;
        }

        $data = array();
        while( $row = mysqli_fetch_assoc($result) ){
        $data[] = $row;
        }
        mysqli_free_result($result);
        return $data;
        }
    }
```

model.php – файл базового класса модели

```
    <?php
    class Model
    {
        public $db; //объект класса БД

        function __construct()
        {
        $this->db = DataBase::getDB();
        }
    }
```

controller.php – файл базового класса контроллера

```
    <?php
    class Controller {
        public $view;//представление
        public $error;//текст ошибки
        public $data;//массив для данных

        function __construct()
        {
        session_start();
        $this->view = new View();
        }

        // действие (action), вызываемое по умолчанию
        function action_index()
        {

        }
    }
```

view.php – файл класса модели

```
    <?php
    class View
    {
        //генерация страницы по шаблону
        function generate($content_view, $template_view, $data = null,$error = null,
$notice = null)
        {
        include 'application/views/'.$template_view;
        }
    //генерация результатов поиска
        function generate_results($results_view, $results_data)
        {
```

```
   include 'application/views/'.$results_view;
   }
}
```

ui.php – вспомогательные функции для создания элементов интерфейса

```
<?php
//создание select по массиву данных
function createSelect($name,$class,$option_data)
{
   echo "<select name='".$name."' class='".$class."'>";
   foreach($option_data as $value=>$text)
   {
   echo "<option value='".$value."'>".$text."</option>";
   }
   echo "</select>";
}
//создание select по массиву данных с группами опций
function createSelectWithGroups($name,$class,$option_data)
{
   echo "<select name='".$name."' class='".$class."'>";
   foreach($option_data as $group=>$options)
   {
   echo "<optgroup label='".$group."'>";
   foreach($options as $value=>$text)
   {
   echo "<option value='".$value."'>".$text."</option>";
   }
   echo "</optgroup>";
   }
   echo "</select>";
}
//создание списка на основе строки с разделителем
function strToList($str,$class,$separator)
{
   $lst = explode($separator, $str);
   echo "<ul class='".$class."'>";
   foreach($lst as $l)
   {
   echo "<li>".$l."</li>";
   }
   echo "</ul>";
}
```

controller_resume.php – контроллер Резюме

```
<?php
class Controller_Resume extends Controller
{
   public $resume; //модель резюме

   function __construct()
   {
   parent::__construct();
   require_once 'application/models/model_resume.php';

   try {
   $this->resume = new Model_Resume();
   } catch (Exception $e) {
   $this->error = $e->getMessage();
   }
```

```
      }
      function action_index()
      {
      if (isset($_GET['id']))
      {
      if ($_SESSION['user_role']=="REPRESENT")
      {
      $resume_id=$_GET['id'];
      try {
      $this->data=$this->resume->getFullResumeOf($resume_id);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if (empty($this->data))
      header("Location: /404");
      }
      else {
      $this->error="Просмотр полного резюме доступен только работодателям.";
      }
      $this->view->generate('resume_view.php', 'templates/main_view.php',$this-
>data,$this->error);
      }
      else
      {
      $this->view->generate('resume_list_view.php',
'templates/main_view.php',null,$this->error);
      }
      }
      //действие поиск по должности
      function action_search()
      {
      try {
      $wantedVacancy=$_POST['wantedVacancy'];
      $this->data=$this->resume->getResumeList($wantedVacancy);
      } catch (Exception $e){
      $this->error = $e->getMessage();
      }
      $this->view->generate_results('resume_list_results.php', $this->data);
      }
   }
```

controller_vacancy.php – контроллер Вакансия

```
   <?php
   class Controller_Vacancy extends Controller
   {
      public $vacancy; //модель вакансия

      function __construct()
      {
      parent::__construct();
      require_once 'application/models/model_vacancy.php';

      try {
      $this->vacancy = new Model_Vacancy();
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      }

      function action_index()
      {
      if (isset($_GET['id']))
```

```
        {
        $vacancy_id=$_GET['id'];
        try {
        $this->data=$this->vacancy->getInfoOf($vacancy_id);
        } catch (Exception $e) {
        $this->error = $e->getMessage();
        }
        if (empty($this->data))
        header("Location: /404");
        $this->view->generate('vacancy_view.php', 'templates/main_view.php',$this-
>data,$this->error);
        }
        else
        {
        $this->view->generate('vacancy_list_view.php',
'templates/main_view.php',null,$this->error);
        }

        }
        //действие поиск по должности
        function action_search()
        {
        try {
        $vacancyName=$_POST['vacancyName'];
        $this->data=$this->vacancy->getList($vacancyName);
        } catch (Exception $e){
        $this->error = $e->getMessage();
        }
        $this->view->generate_results('vacancy_list_results.php', $this->data);
        }
    }
```

controller_404.php – контроллер 404

```
<?php
class Controller_404 extends Controller
{
    function action_index()
    {
    $this->view->generate('404_view.php', 'templates/main_view.php');
    }
}
```

controller_company.php – контроллер компании

```
<?php
class Controller_Company extends Controller
{
    public $company; //модель компания

    function __construct()
    {
    parent::__construct();
    require_once 'application/models/model_company.php';

    try {
    $this->company = new Model_Company();
    } catch (Exception $e) {
    $this->error = $e->getMessage();
    }
    }

    function action_index()
    {
    if (isset($_GET['id']))
```

```
      {
      $company_id=$_GET['id'];
      try {
      $this->data['info']=$this->company->getInfoOf($company_id);
      $this->data['represent']=$this->company-
>getRepresentativeListOf($company_id);
      $this->data['vacancy']=$this->company->getVacancyListOf($company_id);
      $this->data['news']=$this->company->getNewsListOf($company_id);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if (empty($this->data['info'])&&empty($this-
>data['represent'])&&empty($this->data['vacancy'])&&empty($this->data['news']))
      header("Location: /404");
      $this->view->generate('company_view.php', 'templates/main_view.php',$this-
>data,$this->error);
      }
      else
      {
      try {
      $this->data=$this->company->getList();
      } catch (Exception $e){
      $this->error = $e->getMessage();
      }
      $this->view->generate('company_list_view.php', 'templates/main_view.php',
$this->data,$this->error);
      }
      }
   }
```

controller_login.php – контроллер Вход

```
<?php
class Controller_Login extends Controller
{
   public $user; //модель пользователь

   function __construct()
   {
   parent::__construct();
   require_once 'application/models/model_user.php';

   try {
   $this->user = new Model_User();
   } catch (Exception $e) {
   $this->error = $e->getMessage();
   }
   }

   function action_index()
   {
   if (isset($_POST['email'])&&isset($_POST['password']))
   {
   $email=$_POST['email'];
   $password=$_POST['password'];

   try {
   $user_id=$this->user->check($email, $password);
   if ($user_id==0)
   {
   $this->error = "Неверный email или пароль";
   }
   else
```

```
        {
        $this->auth($user_id);
        }
        } catch (Exception $e) {
        $this->error = $e->getMessage();
        }
        }
        $this->view->generate('forms/login_view.php', 'templates/main_view.php',
$this->data,$this->error);
        }
        private function auth($user_id)
        {
        try {
        $_SESSION['user_role']=$this->user->getRoleNameOf($user_id);
        $_SESSION['user_id']=$user_id;
        $avatar=$this->user->getAvatarOf($user_id);
        $_SESSION['user_avatar']="/images/avatars/".$avatar;
        } catch (Exception $e) {
        $this->error = $e->getMessage();
        }
        header("Location: /");
        }
    }
```

controller_404.php – контроллер Выход

```
    <?php
    class Controller_Logout extends Controller
    {
        function action_index()
        {
        session_start();
        session_destroy();
        header("Location: /");
        }
    }
```

controller_main.php – контроллер Главная

```
    <?php
    class Controller_Main extends Controller
    {
        public $resume; //модель резюме
        public $vacancy; //модель вакансия

        function __construct()
        {
        parent::__construct();
        require_once 'application/models/model_resume.php';
        require_once 'application/models/model_vacancy.php';

        try {
        $this->resume = new Model_Resume();
        $this->vacancy = new Model_Vacancy();
        } catch (Exception $e) {
        $this->error = $e->getMessage();
        }
        }

        function action_index()
        {
        try {
        $this->data['resume']=$this->resume->countResume();
        $this->data['vacancy']=$this->vacancy->countVacancy();
        } catch (Exception $e) {
```

```
      $this->error = $e->getMessage();
      }
      $this->view->generate('main_view.php', 'templates/main_view.php',$this-
>data,$this->error);
      }
   }
```

controller_mycompany.php – контроллер Моя компания

```
<?php
class Controller_MyCompany extends Controller
{
   public $user; //модель пользователь
   public $company; //модель компания
   public $represent; //модель представитель

   function __construct()
   {
   parent::__construct();
   require_once 'application/models/model_user.php';
   require_once 'application/models/model_company.php';
   require_once 'application/models/model_represent.php';
   require_once 'application/models/model_news.php';
   require_once 'application/models/model_vacancy.php';
   try {
   $this->user = new Model_User();
   $this->company = new Model_Company();
   $this->represent = new Model_Represent();
   $this->news = new Model_News();
   $this->vacancy = new Model_Vacancy();
   } catch (Exception $e) {
   $this->error = $e->getMessage();
   }
   }
   function action_index()
   {
   $user_id=$_SESSION['user_id'];
   try {
   $company_id=$this->represent->getCompanyIdOf($user_id);
   } catch (Exception $e) {
   $this->error = $e->getMessage();
   }
   if ($company_id==0)
   {
   header("Location: /mycompany/add_company");
   }
   else
   {
   $this->data['buttons']['info_button']['value']="Изменить информацию";
   $this->data['buttons']['info_button']['action']="/mycompany/change_info";
   $this->data['buttons']['vacancy_button']['value']="Добавить вакансию";
   $this->data['buttons']['vacancy_button']['action']="/mycompany/add_vacancy";
   $this->data['buttons']['news_button']['value']="Добавить новость";
   $this->data['buttons']['news_button']['action']="/mycompany/add_news";
   $this->data['info']=$this->company->getInfoOf($company_id);
   $this->data['news']=$this->company->getNewsListOf($company_id);
   $this->data['vacancy']=$this->represent->getVacancyListOf($user_id);

   $this->view->generate('mycompany_view.php', 'templates/main_view.php',$this-
>data,$this->error);
   }
   }
   //действие добавление компании
```

```
	function action_add_company()
	{
	if (!empty($_POST))
	{
	$info=$_POST['info'];
	$phones=$_POST['phones'];
	$name=$_POST['name'];
	$addr=$_POST['addr'];
	$site=$_POST['site'];
	$represent_id=$_SESSION['user_id'];

	try {
	$company_id=$this->represent->addCompany($info,$phones,$name,$addr,$site);
	$this->represent->bindCompany($represent_id, $company_id);
	} catch (Exception $e) {
	$this->error = $e->getMessage();
	}
	if ($this->error=="")
	{
	header("Location: /mycompany");
	}
	}
	$this->view->generate('forms/add_company_view.php',
'templates/main_view.php',$this->data,$this->error);
	}
	//действие добавление вакансии
	function action_add_vacancy()
	{
	if (!empty($_POST))
	{
	$user_id=$_SESSION['user_id'];
	$company_id=$this->represent->getCompanyIdOf($user_id);
	$vacancyName=$_POST['vacancyName'];
	$shelude=$_POST['shelude'];
	$personal=$_POST['personal'];
	$skills=$_POST['skills'];
	$duties=$_POST['duties'];
	$salary=$_POST['salary'];

	try {
	$this->represent->addVacancy($company_id, $vacancyName, $shelude, $personal,
$skills, $duties, $salary, $user_id);
	} catch (Exception $e) {
	$this->error = $e->getMessage();
	}
	if ($this->error=="")
	{
	header("Location: /mycompany");
	}
	}
	$this->view->generate('forms/add_vacancy_view.php',
'templates/main_view.php',$this->data,$this->error);
	}
	//действие добавление новости
	function action_add_news()
	{
	if (!empty($_POST))
	{
	$user_id=$_SESSION['user_id'];
	$company_id=$this->represent->getCompanyIdOf($user_id);
	$subject=$_POST['subject'];
```

```
      $text=$_POST['text'];
      $today=getdate();
      $datePost=$today['year']."-".$today['mon']."-".$today['mday'];
      $dateEnd=$_POST['dateEnd'];
      $short_text=$_POST['short_text'];


      try {
      $this->represent->addNews($company_id, $subject, $text, $datePost, $dateEnd,
$short_text);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /mycompany");
      }
      }
      $this->view->generate('forms/add_news_view.php', 'templates/main_view.php',
$this->data,$this->error);
      }
      //действие изменение новости
      function action_change_news()
      {
      $news_id=$_GET['id'];
      try {
       $this->data=$this->news->getInfoOf($news_id);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if (!empty($_POST))
      {
      $subject=$_POST['subject'];
      $text=$_POST['text'];
      $today=getdate();
      $datePost=$today['year']."-".$today['mon']."-".$today['mday'];
      $dateEnd=$_POST['dateEnd'];
      $short_text=$_POST['short_text'];
      try {
      $this->represent->changeNews($news_id, $subject, $text, $datePost, $dateEnd,
$short_text);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /mycompany");
      }
      }
      $this->view->generate('forms/change_news_view.php',
'templates/main_view.php',$this->data,$this->error);
      }
      //действие изменение вакансии
      function action_change_vacancy()
      {
      $vacancy_id=$_GET['id'];
      try {
       $this->data=$this->vacancy->getInfoOf($vacancy_id);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if (!empty($_POST))
      {
```

```
      $vacancyName=$_POST['vacancyName'];
      $shelude=$_POST['shelude'];
      $personal=$_POST['personal'];
      $skills=$_POST['skills'];
      $duties=$_POST['duties'];
      $salary=$_POST['salary'];
      try {
      $this->represent->changeVacancy($vacancy_id, $vacancyName, $shelude,
$personal, $skills, $duties, $salary);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /mycompany");
      }
      }
      $this->view->generate('forms/change_vacancy_view.php',
'templates/main_view.php',$this->data,$this->error);
      }
   }
```

controller_mypage.php – контроллер Моя страница

```
   <?php
   class Controller_Mypage extends Controller
   {
      public $user; //модель пользователь
      public $student; //модель студент
      public $represent; //модель представитель


      function __construct()
      {
      parent::__construct();
      require_once 'application/models/model_user.php';
      require_once 'application/models/model_student.php';
      require_once 'application/models/model_represent.php';
      require_once 'application/models/model_company.php';


      try {
      $this->user = new Model_User();
      $this->student = new Model_Student();
      $this->represent = new Model_Represent();
      $this->company = new Model_Company();
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      }
      function action_index()
      {
      if (isset($_SESSION['user_role'])&&isset($_SESSION['user_id']))
      {
       try {
      $this->data['info']=$this->user->getInfoOf($_SESSION['user_id']);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      switch ($_SESSION['user_role']) {
      case "STUDENT":
      $this->student();
      break;
      case "REPRESENT":
```

```
      $this->represent();
      break;
      }
      }
      else
      {
      header("Location: /404");
      }
      }
      //действие добавление резюме
      function action_add_resume()
      {
      try {
      $directions=$this->student->getDirections();
      for ($i=0; $i<count($directions); $i++)
      {
      $this->data['directions'][$directions[$i]['type']][$directions[$i]
['direction_id']]=$directions[$i]['name'];
      }
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }


      if (!empty($_POST))
      {
      $user_id=$_SESSION['user_id'];
      $direction_id=$_POST['direction'];
      $personal=$_POST['personal'];
      $skills=$_POST['skills'];
      $wantedVacancy=$_POST['wantedVacancy'];
      $wantedSalary=$_POST['wantedSalary'];
      $wantedShelude=$_POST['wantedShelude'];
      try {
      $this->student->addResume($user_id, $direction_id, $personal, $skills,
$wantedVacancy, $wantedSalary, $wantedShelude);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /mypage");
      }
      }
      $this->view->generate('forms/add_resume_view.php',
'templates/main_view.php',$this->data,$this->error);
      }
   //действие изменение резюме
      function action_change_resume()
      {
      try {
      $directions=$this->student->getDirections();
      $this->data['resume']=$this->student->getResumeOf($_SESSION['user_id']);
      for ($i=0; $i<count($directions); $i++)
      {
      $this->data['directions'][$directions[$i]['type']][$directions[$i]
['direction_id']]=$directions[$i]['name'];
      }
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if (!empty($_POST))
      {
```

```
      $user_id=$_SESSION['user_id'];
      $direction_id=$_POST['direction'];
      $personal=$_POST['personal'];
      $skills=$_POST['skills'];
      $wantedVacancy=$_POST['wantedVacancy'];
      $wantedSalary=$_POST['wantedSalary'];
      $wantedShelude=$_POST['wantedShelude'];
            try {
      $this->student->changeResume($user_id, $direction_id, $personal, $skills,
$wantedVacancy, $wantedSalary, $wantedShelude);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /mypage");
      }
      }
      $this->view->generate('forms/change_resume_view.php',
'templates/main_view.php',$this->data,$this->error);
      }


   //действие изменение инфо студента
      function action_change_info_student()
      {
      try {
      $this->data=$this->user->getInfoOf($_SESSION['user_id']);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }


      if (!empty($_POST))
      {
      $user_id=$_SESSION['user_id'];
      $fio=$_POST['fio'];
      $birthDate=$_POST['birthDate'];
      $email=$_POST['email'];

      try {
      $ex=$this->user->checkEmail($email);
      if($ex=='0' || ($email==$this->data['email']))
      {
      try {
      $this->user->changeInfo($user_id,$fio,$birthDate,$email);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /mypage");
      }
      }
      else
      {
      $this->error = "Пользователь с таким email уже зарегистрирован";
      }
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      }
```

```
      $this->view->generate('forms/change_info_student_view.php',
'templates/main_view.php',$this->data,$this->error);
      }
      //действие изменение инфо представителя
      function action_change_info_represent()
      {
      try {
      $companyList=$this->company->getList();
      $this->data['select']['newCompany']="Компании нет в списке";
      for($i=0; $i<count($companyList); $i++)
      {
      $this->data['select'][$companyList[$i]['company_id']]=$companyList[$i]
['name'];
      }
      $this->data['info']=$this->user->getInfoOf($_SESSION['user_id']);
      $position=$this->represent->getInfoOf($_SESSION['user_id']);
      $this->data['info']['position']=$position['position'];
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }

      if (!empty($_POST))
      {
      $user_id=$_SESSION['user_id'];
      $fio=$_POST['fio'];
      $birthDate=$_POST['birthDate'];
      $email=$_POST['email'];
      $company_id=$_POST['company'];
      $position=$_POST['position'];

      try {
      $ex=$this->user->checkEmail($email);
      if($ex=='0' || ($email==$this->data['info']['email']))
      {
      try {
      $this->user->changeInfo($user_id,$fio,$birthDate,$email);
      $this->represent->changeInfo($user_id,$company_id,$position);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /mypage");
      }
      }
      else
      {
      $this->error = "Пользователь с таким email уже зарегистрирован";
      }
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      }
      $this->view->generate('forms/change_info_represent_view.php',
'templates/main_view.php',$this->data,$this->error);
      }

      private function student()
      {
      $this->data['buttons']['info_button']['value']="Изменить информацию";
```

```
      $this->data['buttons']['info_button']
['action']="/mypage/change_info_student";

      try {
      $this->data['resume']=$this->student->getResumeOf($_SESSION['user_id']);
      if (empty($this->data['resume']))
      {
      $this->data['buttons']['resume_button']['value']="Добавить резюме";
      $this->data['buttons']['resume_button']['action']="/mypage/add_resume";
      }
      else
      {
      $this->data['buttons']['resume_button']['value']="Изменить резюме";
      $this->data['buttons']['resume_button']['action']="/mypage/change_resume";
      }
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      $this->view->generate('mypage/student_view.php', 'templates/main_view.php',
$this->data,$this->error);
      }
      private function represent()
      {
      $this->data['buttons']['info_button']['value']="Изменить информацию";
      $this->data['buttons']['info_button']
['action']="mypage/change_info_represent";


      try {
      $this->data['represent_info']=$this->represent-
>getInfoOf($_SESSION['user_id']);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      $this->view->generate('mypage/represent_view.php',
'templates/main_view.php',$this->data,$this->error);
      }
   }
```

controller_news.php – контроллер Новости

```
   <?php
   class Controller_News extends Controller
   {
      public $news;
      function __construct()
      {
      parent::__construct();
      require_once 'application/models/model_news.php';
      try {
      $this->news = new Model_News();
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      }


      function action_index()
      {
      if (isset($_GET['id']))
      {
      $news_id=$_GET['id'];
      try {
      $this->data=$this->news->getInfoOf($news_id);
      } catch (Exception $e) {
```

```
      $this->error = $e->getMessage();
      }
      if (empty($this->data))
      header("Location: /404");
      $this->view->generate('news_view.php', 'templates/main_view.php',$this-
>data,$this->error);
      }
      else
      {
      try {
      $this->data=$this->news->getActualList();
      } catch (Exception $e){
      $this->error = $e->getMessage();
      }
      $this->view->generate('news_list_view.php', 'templates/main_view.php',$this-
>data,$this->error);
      }
      }
   //действие архив новостей
      function action_archive()
      {
      if (isset($_GET['id']))
      {
      $news_id=$_GET['id'];
      try {
      $this->data=$this->news->getInfoOf($news_id);
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if (empty($this->data))
      header("Location: /404");
$this->view->generate('news_view.php','templates/main_view.php',$this->data,$this-
>error);
      }
      else
      {
      try {
      $this->data=$this->news->getArchiveList();
      } catch (Exception $e){
      $this->error = $e->getMessage();
      }
      $this->view->generate('news_archive_list_view.php',
'templates/main_view.php',$this->data,$this->error);
      }

      }
   }
```

controller_registration.php – контроллер Регистрация

```
<?php
class Controller_Registration extends Controller
{
   public $user;//модель пользователь
   public $company;//модель компания

   function __construct()
   {
   parent::__construct();
   require_once 'application/models/model_company.php';
   require_once 'application/models/model_user.php';

   try {
```

```
	$this->company = new Model_Company();
	$this->user = new Model_User();
	} catch (Exception $e) {
	$this->error = $e->getMessage();
	}
	}
	function action_index()
	{
	try {
	$companyList=$this->company->getList();
	$this->data['select']['newCompany']="Компании нет в списке";
	for($i=0; $i<count($companyList); $i++)
	{
	$this->data['select'][$companyList[$i]['company_id']]=$companyList[$i]
['name'];
	}
	} catch (Exception $e) {
	$this->error = $e->getMessage();
	}
	if (isset($_POST['password'])&&isset($_POST['rep_password']))
	{
	$password=$_POST['password'];
	$rep_password=$_POST['rep_password'];
	if ($password!=$rep_password)
	{
	$this->error = "Пароли должны совпадать";
	}
	else
	{
	if (isset($_POST['email']))
	{
	$email=$_POST['email'];
	try {
	$ex=$this->user->checkEmail($email);
	if($ex=='0')
	{
	$this->register();
	}
	else
	{
	$this->error = "Пользователь с таким email уже зарегистрирован";
	}
	} catch (Exception $e) {
	$this->error = $e->getMessage();
	}
	}
	}
	}
	$this->view->generate('forms/registration_view.php',
'templates/main_view.php',$this->data,$this->error);
	}
	private function register()
	{
	$email=$_POST['email'];
	$password=$_POST['password'];
	$roleName=$_POST['role'];
	$fio=$_POST['fio'];
	$birthDate=$_POST['birthDate'];
	$avatar="default.png";
	$today=getdate();
	$registerDate=$today['year']."-".$today['mon']."-".$today['mday'];
	try {
```

```
      $role_id=$this->user->getRoleID($roleName);
      $user_id=$this->user->add($email, $password, $role_id, $registerDate,
$avatar, $fio, $birthDate);


      if ($roleName=="REPRESENT")
      {
      $company_id=$_POST['company'];
      $position=$_POST['position'];
      if ($company_id=="newCompany")
      {
      $company_id=null;
      }
      $this->user->addRepresent($user_id, $position, $company_id);
      }
      } catch (Exception $e) {
      $this->error = $e->getMessage();
      }
      if ($this->error=="")
      {
      header("Location: /");
      }
      }
   }
```

model_resume.php – модель Резюме

```
   <?php
   class Model_Resume Extends Model
   {
      function __construct()
      {
      parent::__construct();
      }
      //получить список резюме
      function getResumeList($wantedVacancy) {
      $wantedVacancy=$this->db->escape($wantedVacancy);
      $query_str="SELECT user.user_id,fio,wantedVacancy,wantedSalary,wantedShelude
FROM resume LEFT JOIN user ON user.user_id=resume.user_id WHERE wantedVacancy LIKE
'%".$wantedVacancy."%'";
      $data=$this->db->query($query_str);
      return $data;
      }
      //получить резюме студента для представителя
      function getFullResumeOf($id) {
      $id=$this->db->escape($id);
      $query_str="SELECT fio,email,avatar,birthDate,direction.name, personal,
skills, wantedVacancy, wantedSalary, wantedShelude FROM resume LEFT JOIN direction
ON resume.direction_id=direction.direction_id LEFT JOIN user ON
user.user_id=resume.user_id WHERE resume.user_id=".$id;
      $data=$this->db->query($query_str);
      return $data[0];
      }
      //получить число резюме
      function countResume() {
      $query_str="SELECT count(user_id) AS resume_count FROM resume";
      $data=$this->db->query($query_str);
      return $data[0];
      }
   }
```

model_student.php – модель Студент

```
   <?php
   class Model_Student Extends Model
   {
      function __construct()
```

```
      {
      parent::__construct();
      }
      //получить резюме
      function getResumeOf($id) {
      $id=$this->db->escape($id);
      $query_str="SELECT direction.name, personal, skills, wantedVacancy,
wantedSalary, wantedShelude FROM resume LEFT JOIN direction ON
resume.direction_id=direction.direction_id WHERE user_id=".$id;
      $data=$this->db->query($query_str);
      return $data[0];
      }
      //получить направления
      function getDirections() {
      $query_str="SELECT type,direction.direction_id,name FROM direction LEFT JOIN
dir_type ON dir_type.dir_type_id=direction.dir_type_id";
      $data=$this->db->query($query_str);
      return $data;
      }
   //добавить резюме
      function addResume($user_id,$direction_id, $personal, $skills,
$wantedVacancy, $wantedSalary, $wantedShelude) {
      $user_id=$this->db->escape($user_id);
      $direction_id=$this->db->escape($direction_id);
      $personal=$this->db->escape($personal);
      $skills=$this->db->escape($skills);
      $wantedVacancy=$this->db->escape($wantedVacancy);
      $wantedSalary=$this->db->escape($wantedSalary);
      $wantedShelude=$this->db->escape($wantedShelude);
      $query_str="INSERT INTO resume(user_id, direction_id, personal, skills,
wantedVacancy, wantedSalary, wantedShelude) VALUES (".$user_id.",".
$direction_id.",'".$personal."','".$skills."','".$wantedVacancy."','".
$wantedSalary."','".$wantedShelude."')";
      $this->db->query($query_str);
      $id=$this->db->ins_id();
      return $id;
      }
   //изменить резюме
      function changeResume($user_id,$direction_id, $personal, $skills,
$wantedVacancy, $wantedSalary, $wantedShelude) {
      $user_id=$this->db->escape($user_id);
      $direction_id=$this->db->escape($direction_id);
      $personal=$this->db->escape($personal);
      $skills=$this->db->escape($skills);
      $wantedVacancy=$this->db->escape($wantedVacancy);
      $wantedSalary=$this->db->escape($wantedSalary);
      $wantedShelude=$this->db->escape($wantedShelude);
      $query_str="UPDATE resume SET direction_id=".$direction_id.", personal='".
$personal."', skills='".$skills."', wantedVacancy='".$wantedVacancy."',
wantedSalary='".$wantedSalary."', wantedShelude='".$wantedShelude."' WHERE
user_id=".user_id;
      $this->db->query($query_str);
      $id=$this->db->ins_id();
      return $id;
      }
   }
```

model_user.php – модель Пользователь

```
<?php
class Model_User Extends Model
{
   function __construct()
   {
```

```
      parent::__construct();
      }
   //получить инфо
      function getInfoOf($id) {
      $id=$this->db->escape($id);
      $query_str="SELECT * FROM user WHERE user_id=".$id;
      $data=$this->db->query($query_str);
      return $data[0];
      }
   //добавить пользователя
      function add($email, $password, $role_id, $registerDate, $avatar, $fio,
$birthDate) {
      $email=$this->db->escape($email);
      $password=$this->db->escape($password);
      $role_id=$this->db->escape($role_id);
      $registerDate=$this->db->escape($registerDate);
      $avatar=$this->db->escape($avatar);
      $fio=$this->db->escape($fio);
      $birthDate=$this->db->escape($birthDate);
      $query_str="INSERT INTO user(user_id, email, password, role_id,
registerDate, avatar, fio, birthDate) VALUES (NULL,'".$email."','".
$password."','".$role_id."','".$registerDate."','".$avatar."','".$fio."','".
$birthDate."')";
      $this->db->query($query_str);
      $id=$this->db->ins_id();
      return $id;
      }
   //добавить представителя
      function addRepresent($user_id, $position, $company_id) {
      $user_id=$this->db->escape($user_id);
      $company_id=$this->db->escape($company_id);
      $query_str="INSERT INTO represent(user_id, position, company_id) VALUES ('".
$user_id."','".$position."','".$company_id."')";
      $this->db->query($query_str);
      $id=$this->db->ins_id();
      return $id;
      }
   //проверить почту
      function checkEmail($email) {
      $email=$this->db->escape($email);
      $query_str="SELECT COUNT(user_id) AS C FROM user WHERE email='".$email."'";
      $data=$this->db->query($query_str);
      return $data[0]['C'];
      }
   //проверить почту и пароль
      function check($email,$password) {
      $email=$this->db->escape($email);
      $password=$this->db->escape($password);
      $query_str="SELECT user_id FROM user WHERE email='".$email."' AND
password='".$password."'";
      $data=$this->db->query($query_str);
      if (empty($data))
      {
      $data[0]['user_id']=0;
      }
      return $data[0]['user_id'];
      }
   //получить ID роли
      function getRoleID($roleName) {
      $roleName=$this->db->escape($roleName);
      $query_str="SELECT role_id FROM role WHERE roleName='".$roleName."'";
      $data=$this->db->query($query_str);
```

```
      return $data[0]['role_id'];
      }
   //получить имя роли пользователя
      function getRoleNameOf($user_id) {
      $user_id=$this->db->escape($user_id);
      $query_str="SELECT roleName FROM user LEFT JOIN role ON user.role_id =
role.role_id WHERE user_id ='".$user_id."'";
      $data=$this->db->query($query_str);
      return $data[0]['roleName'];
      }
   //получить аватар
      function getAvatarOf($user_id) {
      $user_id=$this->db->escape($user_id);
      $query_str="SELECT avatar FROM user WHERE user_id ='".$user_id."'";
      $data=$this->db->query($query_str);
      return $data[0]['avatar'];
      }
   //изменить инфо
      function changeInfo($user_id, $fio, $birthDate, $email) {
      $user_id=$this->db->escape($user_id);
      $email=$this->db->escape($email);
      $password=$this->db->escape($password);
      $fio=$this->db->escape($fio);
      $birthDate=$this->db->escape($birthDate);
      $query_str="UPDATE user SET email='".$email."',fio='".$fio."',birthDate='".
$birthDate."' WHERE user_id=".$user_id;
      $this->db->query($query_str);
      }
   }
```

model_vacancy.php – модель Вакансия

```
   <?php
   class Model_Vacancy Extends Model
   {
      function __construct()
      {
      parent::__construct();
      }
   //получить список вакансий
      function getList($vacancyName) {
      $vacancyName=$this->db->escape($vacancyName);
      $query_str="SELECT * FROM vacancy LEFT JOIN company ON
vacancy.company_id=company.company_id WHERE vacancyName LIKE '%".
$vacancyName."%'";
      $data=$this->db->query($query_str);
      return $data;
      }
   //получить вакансию
      function getInfoOf($id) {
      $id=$this->db->escape($id);
      $query_str="SELECT * FROM vacancy WHERE vacancy_id=".$id;
      $data=$this->db->query($query_str);
      return $data[0];
      }
   //получить число вакансий
      function countVacancy() {
      $query_str="SELECT count(vacancy_id) AS vacancy_count FROM vacancy";
      $data=$this->db->query($query_str);
      return $data[0];
      }
   }
```

model_company.php – модель Компания

```
   <?php
```

```
class Model_Company Extends Model
{
   function __construct()
   {
   parent::__construct();
   }
   function getList() {
   $query_str="SELECT company_id,name FROM company ORDER BY name";
   $data=$this->db->query($query_str);
   return $data;
   }
//получить инфо компании
   function getInfoOf($id) {
   $id=$this->db->escape($id);
   $query_str="SELECT * FROM company WHERE company_id=".$id;
   $data=$this->db->query($query_str);
   return $data[0];
   }
//получить список представителей компании
   function getRepresentativeListOf($id) {
   $id=$this->db->escape($id);
   $query_str="SELECT represent.user_id,position,fio FROM represent LEFT JOIN
user ON user.user_id=represent.user_id WHERE company_id=".$id;
   $data=$this->db->query($query_str);
   return $data;
   }
//получить список вакансий компании
   function getVacancyListOf($id) {
   $id=$this->db->escape($id);
   $query_str="SELECT * FROM vacancy WHERE company_id=".$id;
   $data=$this->db->query($query_str);
   return $data;
   }
//получить список новостей компании
   function getNewsListOf($id) {
   $id=$this->db->escape($id);
   $query_str="SELECT * FROM news WHERE company_id=".$id;
   $data=$this->db->query($query_str);
   return $data;
   }
}
```

model_news.php – модель Новости

```
<?php
class Model_News Extends Model
{
   function __construct()
   {
   parent::__construct();
   }
//получить список актуальных новостей
   function getActualList() {
   $query_str="SELECT * FROM news LEFT JOIN company ON
news.company_id=company.company_id WHERE dateEnd>=CURDATE()";
   $data=$this->db->query($query_str);
   return $data;
   }
//поличить список архивных новостей
   function getArchiveList() {
   $query_str="SELECT * FROM news LEFT JOIN company ON
news.company_id=company.company_id WHERE dateEnd<CURDATE()";
   $data=$this->db->query($query_str);
   return $data;
```

```
    }
  //получить новость
    function getInfoOf($id) {
    $id=$this->db->escape($id);
    $query_str="SELECT * FROM news LEFT JOIN company ON
news.company_id=company.company_id WHERE news_id=".$id;
    $data=$this->db->query($query_str);
    return $data[0];
    }
  }
```

model_represent.php – модель Представитель

```
  <?php
  class Model_Represent Extends Model
  {
    function __construct()
    {
    parent::__construct();
    }
  //получить инфо представителя
    function getInfoOf($id) {
    $id=$this->db->escape($id);
    $query_str="SELECT company.name,position FROM represent LEFT JOIN company ON
company.company_id=represent.company_id WHERE user_id=".$id;
    $data=$this->db->query($query_str);
    return $data[0];
    }
  //получить ID компании представителя
    function getCompanyIdOf($id) {
    $id=$this->db->escape($id);
    $query_str="SELECT company_id FROM represent WHERE user_id=".$id;
    $data=$this->db->query($query_str);
    return $data[0]['company_id'];
    }
  //добавить компанию
    function addCompany($info,$phones,$name,$addr,$site) {
    $info=$this->db->escape($info);
    $phones=$this->db->escape($phones);
    $name=$this->db->escape($name);
    $query_str="INSERT INTO company(company_id, info, phones, name,addr,site)
VALUES (NULL,'".$info."','".$phones."','".$name."','".$addr."','".$site."')";
    $this->db->query($query_str);
    $id=$this->db->ins_id();
    return $id;
    }
  //привязать компанию к представителю
    function bindCompany($represent_id,$company_id) {
    $represent_id=$this->db->escape($represent_id);
    $company_id=$this->db->escape($company_id);
    $query_str="UPDATE represent SET company_id='".$company_id."' WHERE
user_id=".$represent_id;
    $this->db->query($query_str);
    }
  //добавить новость
    function addNews($company_id, $subject, $text, $datePost, $dateEnd,
$short_text) {
    $company_id=$this->db->escape($company_id);
    $subject=$this->db->escape($subject);
    $text=$this->db->escape($text);
    $datePost=$this->db->escape($datePost);
    $dateEnd=$this->db->escape($dateEnd);
```

```
      $short_text=$this->db->escape($short_text);
      $query_str="INSERT INTO news(news_id, company_id, subject, text, datePost,
dateEnd, short_text) VALUES (NULL,'".$company_id."','".$subject."','".$text."','".
$datePost."','".$dateEnd."','".$short_text."')";
      $this->db->query($query_str);
      $id=$this->db->ins_id();
      return $id;
      }
   //добавить вакансию
      function addVacancy($company_id, $vacancyName, $shelude, $personal, $skills,
$duties, $salary, $user_id) {
      $company_id=$this->db->escape($company_id);
      $vacancyName=$this->db->escape($vacancyName);
      $shelude=$this->db->escape($shelude);
      $personal=$this->db->escape($personal);
      $skills=$this->db->escape($skills);
      $duties=$this->db->escape($duties);
      $salary=$this->db->escape($salary);
      $user_id=$this->db->escape($user_id);
      $query_str="INSERT INTO vacancy(vacancy_id, company_id, vacancyName,
shelude, personal, skills, duties, salary, user_id) VALUES (NULL,'".
$company_id."','".$vacancyName."','".$shelude."','".$personal."','".$skills."','".
$duties."','".$salary."','".$user_id."')";
      $this->db->query($query_str);
      $id=$this->db->ins_id();
      return $id;
      }
   //получить список вакансий представителя
      function getVacancyListOf($id) {
      $id=$this->db->escape($id);
      $query_str="SELECT * FROM vacancy WHERE user_id=".$id;
      $data=$this->db->query($query_str);
      return $data;
      }
   //изменить новость
      function changeNews($news_id, $subject, $text, $datePost, $dateEnd,
$short_text) {
      $news_id=$this->db->escape($news_id);
      $subject=$this->db->escape($subject);
      $text=$this->db->escape($text);
      $datePost=$this->db->escape($datePost);
      $dateEnd=$this->db->escape($dateEnd);
      $short_text=$this->db->escape($short_text);
      $query_str="UPDATE news SET subject='".$subject."', text='".$text."',
datePost='".$datePost."', dateEnd='".$dateEnd."', short_text='".$short_text."'
WHERE news_id=".$news_id;
      $this->db->query($query_str);
      }
   //изменить вакансию
      function changeVacancy($vacancy_id, $vacancyName, $shelude, $personal,
$skills, $duties, $salary) {
      $vacancyName=$this->db->escape($vacancyName);
      $shelude=$this->db->escape($shelude);
      $personal=$this->db->escape($personal);
      $skills=$this->db->escape($skills);
      $duties=$this->db->escape($duties);
      $salary=$this->db->escape($salary);
      $vacancy_id=$this->db->escape($vacancy_id);
      $query_str="UPDATE vacancy SET vacancyName='".$vacancyName."', shelude='".
$shelude."', personal='".$personal."', skills='".$skills."', duties='".$duties."',
salary='".$salary."' WHERE vacancy_id=".$vacancy_id;
      $this->db->query($query_str);
```

```
      }
   //изменить инфо представителя
      function changeInfo($represent_id,$company_id,$position) {
      $represent_id=$this->db->escape($represent_id);
      $company_id=$this->db->escape($company_id);
      $position=$this->db->escape($position);
      $query_str="UPDATE represent SET company_id='".$company_id."', position='".
$position."' WHERE user_id=".$represent_id;
      $data=$this->db->query($query_str);
      }
   }
```