Министерство просвещения Российской Федерации
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»

Институт математики, физики, информатики и технологий

Кафедра информатики, информационных технологий
и методики обучения информатике

# ИНФОРМАЦИОННАЯ СИСТЕМА ДЛЯ СЛУЖБЫ ЗАНЯТОСТИ

*Выпускная квалификационная работа*
*бакалавра по направлению подготовки*
*09.03.02 – Информационные системы и технологии*

Работа допущена к защите
«____»_____ 2024 г.
Зав. кафедрой _____

Исполнитель: студент группы ИСиТ 1931z
Замараев В.В.

Руководитель: кандидат педагогических наук,
доцент кафедры ИИТ и МОИ
Газейкина А.И.

Екатеринбург – 2024

# Реферат

Замараев В.В.ИНФОРМАЦИОННАЯ СИСТЕМА ДЛЯ СЛУЖБЫ ЗАНЯТОСТИ, выпускная квалификационная работа: 73 стр., рис 66, табл. 3, библ. 35 назв., приложений 6.

*Ключевые слова*: ИНФОРМАЦИОННАЯ СИСТЕМА, СЛУЖБА ЗАНЯТОСТИ, PYTHON, DJANGO.

*Объект разработки* – информационная система для службы занятости.

*Цель работы* – разработка информационной системы центра занятости, обеспечивающей возможность для пользователей добавлять в систему резюме или вакансии, в зависимости от типа учётной записи.

В работе описаны результаты проектирования и разработки информационной системы, обеспечивающих формирование контента сайта в автоматизированном режиме на основании анкет, заполненных зарегистрированными пользователями системы.

Система реализована в среде разработки Visual Studio Code с использованием языка Python и фреймворка Django.

Система может быть использована :для добавления резюме (для пользователей, которые ищут для себя вакансию), для добавления вакансий (для пользователей, которые предлагают работу) и для просмотра, как резюме, так и вакансий. Добавлять публикации на сайт могут только зарегистрированные пользователи, право на просмотр контента сайта есть абсолютно у всех пользователей, вне зависимости зарегистрированы они или нет.

**Оглавление**

# Введение

В современном мире существует огромное количество программных продуктов, созданных для применения практически во всех областях предпринимательской деятельности. Это информационные системы, которые специализируются на работе в сферах с определённым профилем деятельности. Данный класс программного обеспечения позволяет поднимать качество работы с пользователями и результативность работы организации в целом за счёт автоматизации большинства операций.

Кадровый бизнес, в силу своих определённых особенностей, развивается медленнее других отраслей, в которых процесс автоматизации начался раньше в связи со своей большей развитости и своей большей финансовой обеспеченностью. Но существуют ощутимые различия в автоматизации предприятия занимающегося производством или торгового предприятия и центра занятости. Фундамент этих отличий заключается в том, что рекрутинг - это по большей части информационный бизнес. Соответственно, любые иновации в информационных технологиях отражаются на рекрутинге. Процедура сбора данных и обмена информацией на сегодняшний день не возможен без использования электронных ресурсов.

Сфера рекрутинга использует новейшие информационные сервисы, которые предлагает сфера информационных технологий. Современная информационная система для службы занятости должна выполнять:

- огромное количество рутинной работы, связанной с обработкой резюме, упорядочиванием и поиском по базе резюме;
- уменьшить загруженность сотрудников службы занятости;
- исключить неудобство бумажного документооборота (возможность потери важных документов, а также потеря времени на поиск необходимого документа и т.п.);

Проблемы, существующие на данный момент, можно классифицировать следующим образом:

Технические трудности, которые в большинстве своём связаны с применением бумажных носителей, которые уже давно устарели, в качестве основных носителей информации. А также ограниченное применение возможностей, которые предоставляет персональный компьютер;

Организационные трудности, которые напрямую связаны с устаревшими и не оптимизированными способами обработки информации о претендентах, применяемыми в текущий момент времени;

Социальные проблемы, которые связаны с рутинным характером операций, связанных с поиском персонала и обработкой анкет.

Цель работы – разработать информационную систему для службы занятости.

Задачи:

1. Обосновать актуальность информационной системы для службы занятости.
2. Проанализировать технологии и методы для разработки информационной системы для службы занятости.
3. Подготовить техническое задание на разработку информационной системы для службы занятости.
4. Спроектировать и реализовать информационную систему для службы занятости.
5. Провести апробацию информационной системы методом экспертных оценок.

# Глава 1. Теоретические основы разработки информационной системы для службы занятости

## 1.1 Обоснование актуальности разрабатываемой информационной системы службы занятости

Вследствие того, что в мире в последнее время складывается непростая ситуация в экономике, которая в свою очередь проецируется на отечественный рынок труда. Появилась острая необходимость импортозамещения в огромном количестве отраслей отечественной экономики. И соответственно значительно возросла необходимость в центрах занятости. Также в сегодняшнем мире на рынке труда присутствует огромное количество предприятий, которые постоянно расширяются, существует необходимость постоянного подбора нового персонала.

Для увеличения эффективности работы, имеет смысл автоматизировать выполнение рутинных операций посредством информационной системы. Тем самым минимизировать нагрузку на консультантов, которые сортируют резюме, размещают информацию на сайте центра и выполняют другие операции, с которыми вполне может справиться информационная система. Уровень информационных технологий на сегодняшний день позволяет достаточно просто и эффективно найти решение существующей проблемы посредством применения информационных систем.

Информационная система может позволить повысить управляемость бизнеса за счёт автоматизации процессов и предоставлению пользователю агентства релевантных данных о состоянии дел на рынке вакансий в любой момент времени.

Такого рода система позволит автоматизировать и в разы ускорить время обработки рутинных операций. Использование в системе методов, позволяю-

щих автоматизировать задачу подбора персонала, значительно повысит эффективность работы центра занятости.

Заказчики, как правило, всегда претендуют на высокое качество поиска высококлассных специалистов в короткие сроки. Применение обычных методов оптимизации в данном случае неэффективно. В лучшем случае оптимизация работы экспертов может позволить добиться ускорения операций применяемых для поиска информации, но, ни в коем случае никак не повысит качество поиска, потому как при этом нельзя исключить человеческого фактора и крайне сложно, а порой и невозможно, в короткие сроки на должном уровне оценить квалификацию претендента.

Уровень современных информационных технологий способствует простому и эффективному решению существующих проблем посредством использования специализированных информационных систем.

Уже достаточно долго стоит вопрос о применении автоматизированных информационных систем для центров занятости, и особенно он актуален в свете текущей ситуации в экономике. В современном обществе, где доминирующее положение любой организации, включая центры занятости, определяются, прежде всего, его возможностями по хранению, доступу, а также качественной обработки информации. Огромнейшее значение приобретает квалифицированное применение ведущих достижений в области информационных технологий.

На сегодняшний день на рынке представлено огромное количество сайтов занимающихся освещением трудового рынка. В одной из публикаций на сайте naim.ru имеется обзор на один из таких сервисов, занимающих лидирующие позиции на рынке труда – это сайт rabota.ru.

«Rabota.ru – это один самых посещаемых ресурсов по трудоустройству в русскоязычном сегменте глобальной сети. Успех популярности этого сайта заключается в продуманном до мелочей интерфейсе и системой управления личным кабинетом. В отличие от большинства конкурирующих с «работа. ру» ре-

сурсов, здесь видна подробная статистика по категориям. Каждый пользователь может ознакомиться со списком лучших работодателей и самых востребованных вакансий недели.

Также хотелось бы отметить «чистоту» данного ресурса. Система спам-фильтра позволяет удалять подозрительные и несоответствующие правилам вакансии и резюме, что позволяет уберечь потенциальных соискателей от возможного обмана. К тому же, на сайте работа.ру постоянно производится удаление устаревших записей, чтобы предоставить пользователям актуальные данные как со стороны работодателя, так и со стороны тех, кто размещает своё резюме.

Заканчивая обзор данного ресурса по трудоустройству, хотелось бы обратить внимание на один недостаток. Огромное количество рубрик, каталогов и всяческих публикаций приводит к тому, что посетитель начинает слегка «теряться», что не есть хорошо для сайта, основанного на принципе точного и быстрого поиска работы»[1].

После регистрации на данном ресурсе появляется возможность добавить резюме. Также имеется возможность подписаться на определённые вакансии, при нет возможности добавить свою вакансию. Не предусмотрены возможности добавления изображения и комментариев к публикациям.

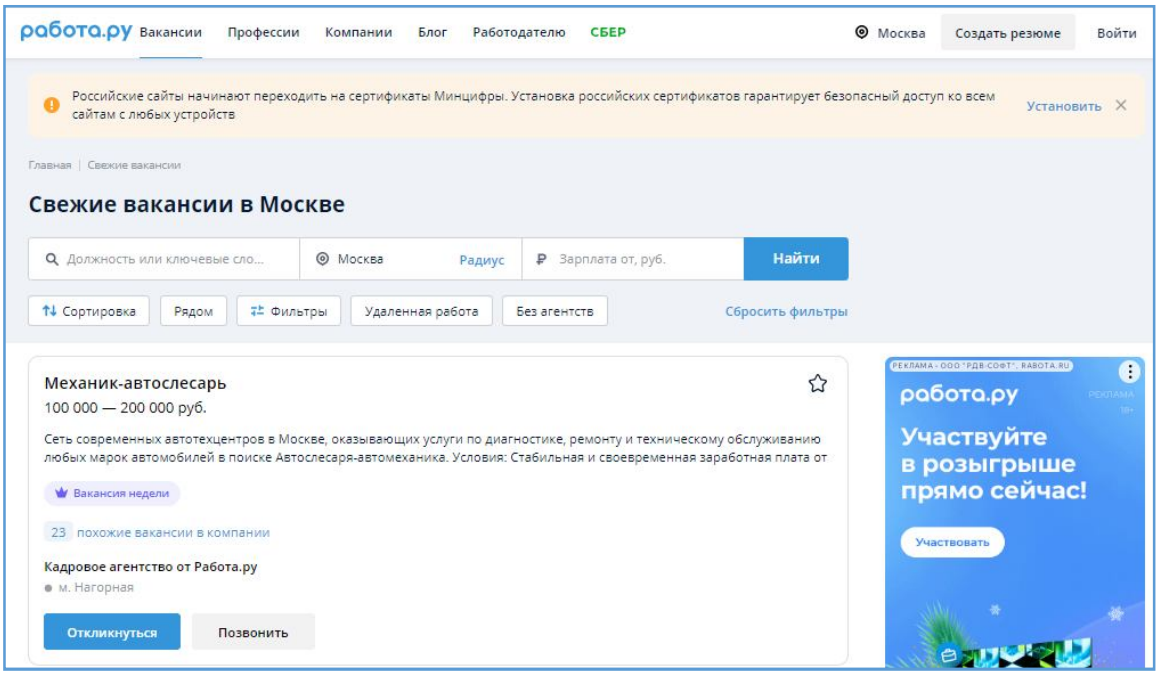



*Рисунок 1. Главная страница сайта «Работа.ру»*

Следующий ресурс, который будет рассмотрен это ir-center.ru. Этот сервис объединяет внутри себя все центры занятости Российской Федерации. На данный момент не представлены только два региона России – это Московская область и Крым.

На этом ресурсе присутствует возможность получить информацию о наличии вакансий в различных регионах РФ. Размещение объявлений на сайте «ir-center.ru» предусмотрено только через региональные центры занятости. Для размещения объявления регистрация на сайте не требуется. В связи с этим можно утверждать, что дальнейшее управление своими объявлениями невозможно, так как отсутствует привязка к конкретному пользователю.

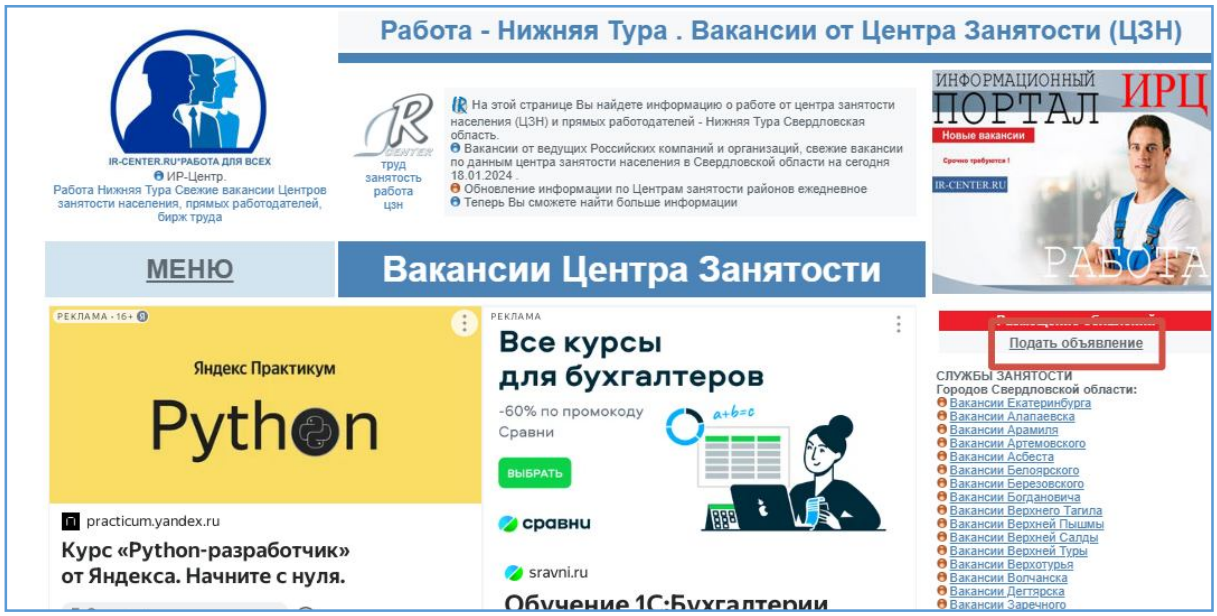



*Рисунок 2. Главная страница сайта ir-center.ru (обзор меню).*

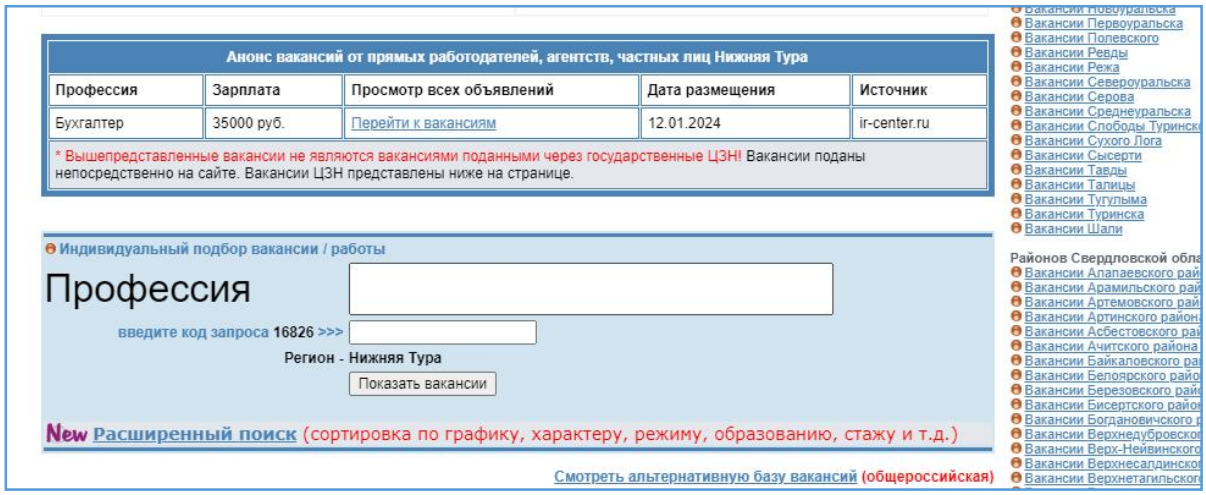



*Рисунок 3. Главная страница сайта ir-center.ru (обзор блока публикаций).*

Следующий ресурс, приведённый в качестве примера это cnz-gorod.ru. Этот ресурс берёт актуальные данные с рассмотренного выше сайта «ir-center.ru». На данном сайте не предусмотрена возможность каких-либо публикаций. Этот ресурс приведён лишь как доказательство того, что такие сайты тоже существуют.

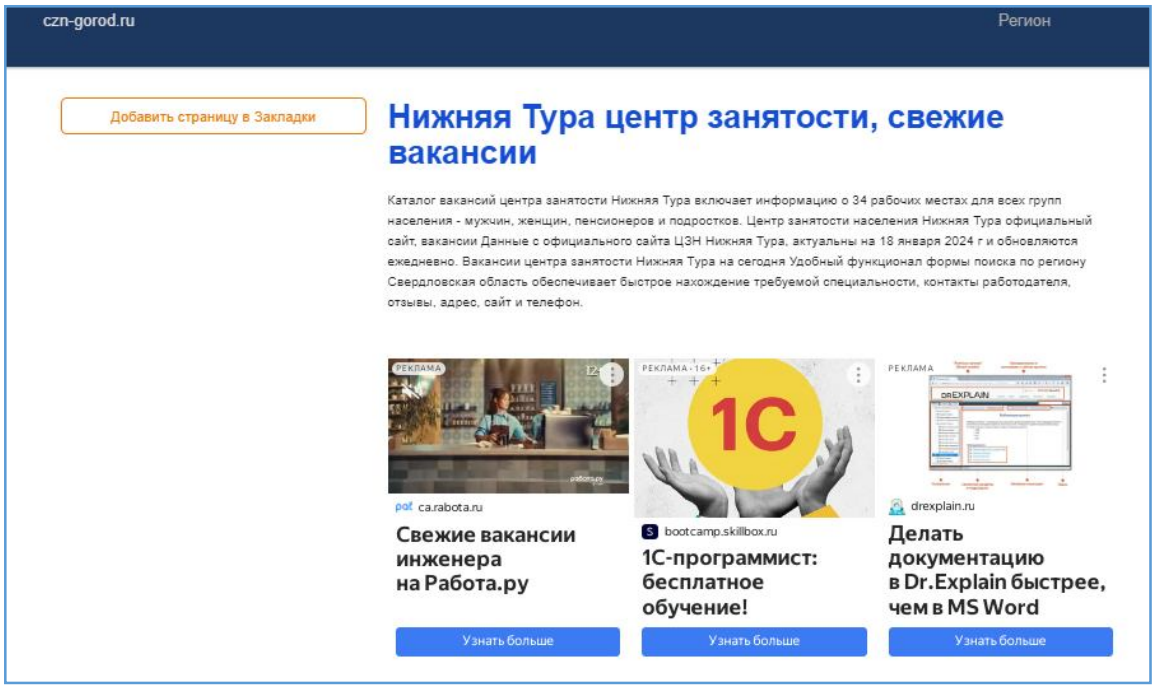



*Рисунок 4. Обзор главной страницы сайта cnz-gorod.ru(часть 1).*

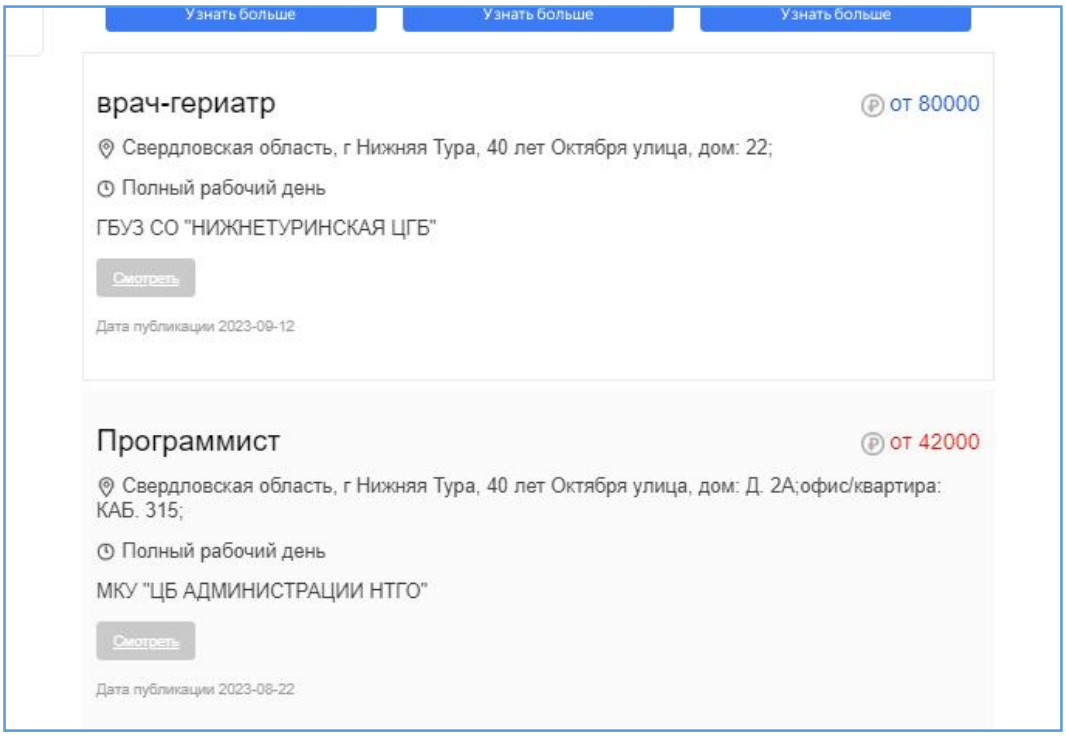



*Рисунок 5. Обзор главной страницы сайта cnz-gorod.ru(часть 2).*

Подводя итого всего вышеописанного, заносим итоговые данные в таблицу. В таблице отображены функциональные критерии разрабатываемой системы для сравнения с сервисами описанными выше.

*Таблица 1.*

*Анализ существующих информационных систем*

| **Функциональные критерии** | **Работа.ру** | **Ir-center.ru** | **Czn-gorod.ru** | **Разрабатываемая система** |
|---|---|---|---|---|
| Регистрация | Да | Нет | Нет | Да |
| Профиль пользователя | Да | Нет | Нет | Да |
| Добавление резюме | Да | Нет | Нет | Да |
| Добавление вакансии | Нет | Да | Нет | Да |
| Добавление изображения | Нет | Нет | Нет | Да |
| Добавление дополнительных изображений | Нет | Нет | Нет | Да |
| Поиск по сайту | Да | Да | Да | Да |
| Комментарии | Нет | Нет | Нет | Да |
| Просмотр вакансий\резюме | Да | Да | Да | Да |

Таким образом, каждый сервис имеет свои преимущества и недостатки. И очевидно, что идеальных информационных систем для центра занятости на сегодняшний день нет. Таким образом, задача проектирования новой информационной системы для центра занятости может считаться актуальной.

## 1.2 Выбор технологий, методов и инструментов разработки информационной системы службы занятости

Формирование моделей бизнес-процессов – это одно из самых быстро развивающихся течений системного анализа. Но, невзирая на то, что имеет ме-

сто большое количество методик и инструментов для моделирования бизнес - процессов, не существует единых стандартов для определения их качества.

Единое стремление к росту потребностей на услуги консультантов в области информационных технологий порождается тем, что бизнес становится с каждым днём всё сложнее, а условия его развития - неоднозначные. С другой точки зрения, в анализируемой области отмечается дефицит высококлассных специалистов. Проблема усложняется еще и тем, что выучить такого профессионала в стенах какого-либо учебного заведения практически невозможно: кроме каких-то теоретических знаний также нужен опыт работы в проектах, который приобретается с годами.

Без сомнений любая модель это лишь визуализация нашего опыта о конкретном процессе. Мы делаем модель состояния процесса, то есть передаём алгоритм действий и изменений, которые реально происходят на объекте.

Любой процесс представляет собой алгоритм действий, но это алгоритм может иметь разветвлённые пути развития. Именно поэтому надо принимать во внимание самые разнообразные ситуации, в том числе те, которые могут приводить к разветвлению процессов.

На сегодняшний день данные статистики благополучности ведения проектов связанных с автоматизацией оставляют желать лучшего, так как всего лишь 25 процентов всех подобных проектов завершаются успехом. Вместе с тем, необходимым условием благополучия развития, работы и конкурентоспособности каждого большого предприятия считается существование корпоративной информационной системы.

Современные технологии позволяют очень быстро внедрять информационные системы по подготовленным требованиям. Но достаточно часто случается так, что эти системы не устраивают заказчиков. Главной причиной такой ситуации является неточная, неправильная или недостаточно полная формулировка требований к информационной системе. Сложность формирования тре-

бований к информационной системе является на сегодняшний день одной из самых трудно формализуемых и особенно дорогих и трудных для корректировки при совершении ошибки. Поэтому очень важна роль ранних этапов жизненного цикла создания информационной системы, когда эти требования обязательно нужно выявить и формализовать.

Именно поэтому для построения правильной информационной системы необходима технология, которая бы могла сгенерировать требования к информационной системе, создать проект и разработать саму систему, соответствующую этим требованиям. Существование такой технологии считается самым важным фактором успеха при создании информационной системы.

В связи с огромным количеством различных предприятий в современном мире, а также их расширением, имеют место проблемы подбора персонала и управления им. На своем жизненном пути у большинства компаний рано или поздно появляется необходимость обращения к центрам занятости.

Автоматизация даёт возможность повышения эффективности работы за счет сокращения времени выполнения повседневных задач.

Подобная система помогает повысить управляемость процессов за счет их формализации и создания единых стандартов для работы, за счет предоставления всем пользователям системы наглядной информации по любому интересующему запросу.

Использование в системе методов, которые позволят автоматизировать процесс подбора персонала, значительно повысит эффективность деятельности центра занятости.

Таким образом, задача проектирования новых информационных систем для центра занятости может считаться актуальной.

***Выбор CASE-средств моделирования***

Для аналитики CASE-средств моделирования информационных систем была использована статья «Выбор CASE-средств» размещённая на web-сайте studfile.net[2].

Для удобства сравнения результатов проведённой аналитики программных продуктов итоги анализа сведены в таблицу (см. таблицу 2) ».

*Таблица 2*

*Сравнительный анализ CASE-средств*

| Параметры сравнения | Rational Rose Modeler | Telelogic Tau Modeler | MicrosoftVisio 2007 | (BPWin) |
|---|---|---|---|---|
| Платформа | Windows/Unix | Windows | Windows | Windows |
| Системные требования | CPU не менее 800 МГц. RAM от 1 Гб. Свободного дискового пространства 2 Гб. | CPU не менее 500 МГц.RAM от 256 Мб. Свободного дискового пространства 150 Мб. | CPU не менее 600 МГц.RAM от 256 Мб. Свободного дискового пространства 300 Мб. | CPU не менее 500 МГц. RAM от 256 Мб. Свободного дискового пространства 100 Мб. |
| Поддержка методологий | UML 2.0 | UML 2.0 | UML 2.0 | IDEF0, DFD, IDEF3 |

Следовательно, для создания моделей согласно структурному подходу CASE-средство в нашем конкретном случае подходит продукт**All Fusion Process Modeler (BPwin).** Этот продукт больше всего подходит для реализации нашей задачи.

**Выбор технологии для реализации информационной системы центра занятости**

Информационные технологии это достаточно широкая отрасль, и для того чтобы поставить задачу для реализации информационной системы центра занятости необходимо обратиться к нормативным документам регламентирующим требования к качеству программного обеспечения. В этой работе при-

мем за основу документ: ГОСТ Р ИСО/МЭК 9126-93 «Оценка программной продукции. Характеристики качества и руководства по их применению» [3].

Самыми ключевыми параметрами, в соответствии с задачами которые планируется возложить на разрабатываемую систему, являются: надёжность, эффективность и функциональность системы. Значимым параметром является удобство использования, а так же совместимость и переносимость системы.

**Надёжность** – это уровень выполнения информационной системой, при указанных условиях определённых функций изначально заложенных в систему в течении всего срока эксплуатации. Под надёжностью также подразумеваются следующие определения:

*Готовность* -уровень работоспособности и доступности системы.

*Завершённость* – соответствие продукта требованиям надёжности.

*Конфиденциальность* – обеспечение информационной системой ограничение доступа к информации только для тех, кому подобный доступ предоставлен.

*Сопровождаемость, модифицируемость* - эффективность и результативность с которыми система может быть модифицирована специалистами по её обслуживанию.

**Результативность, эффективность** – точность и, с которой пользователь достигает поставленных задач.

**Функциональная пригодность** – степень в которой система гарантирует функционирование в соответствии с изначально заявленными потребностями. Под функциональностью также подразумеваются следующие определения.

*Функциональная целесообразность* – степень функционального упрощения достижения целей и выполнения определённых задач.

*Функциональная полнота* – степень покрытия комплексом функций всех конкретных задач и целей пользователя.

**Удобство использования (удовлетворённость)** - способность системы удовлетворять требованиям пользователя в указанных условиях использования продукта. Под удобством использования также предполагаются следующие пункты:

*Полноценность*–степень удовлетворённости пользователя достижением поставленных задач.

*Комфорт* – уровень удовлетворённости пользователя физическим комфортом от использования системы.

*Доверие* – степень уверенности пользователя в том, что информационная система осуществляет свои функции, так как предполагалось изначально.

**Переносимость, мобильность** – степень понятности эффективного и рационального переноса системы из одной среды (операционной системы, программного обеспечения) в другую. Также к мобильности и переносимости относятся:

*Адаптируемость* – степень понятности эффективной и рациональной адаптации для усовершенствованных или отличающихся программного обеспечения, аппаратных средств, других операционных систем.

*Взаимозаменяемость* – способность системы заменить другой программный продукт для решения тех же задач.

Исходя из изложенного выше учитывая требования к функционалу информационной системы центра занятости, можно сделать определённые выводы. Оптимальным вариантом для реализации разрабатываемой системы является использование веб - технологий. Именно этот вариант отвечает всем заявленным требованиям. В качестве обоснования актуальности использования этих технологий сопоставим критерии, приведённые выше с разрабатываемой информационной системой.

При использовании веб - технологий проблема переносимости и мобильности информационной системы решается сама собой. Пользователь может ис-

пользовать данную информационную систему на любом устройстве. Главное условие наличие на этом устройстве браузера. Причём не важно, какая операционная система установлена на этом устройстве.

Также в отличие от приложений, установленных на компьютере или на каком либо мобильном устройстве, при необходимости внесения в систему каких либо изменений не потребуется обновлять или тем более переустанавливать приложение. Все изменения происходят на сервере и в большинстве случаев пользователи не замечают, когда это происходит.

Что касается остальных критериев, то они являются общими для разработки любого программного обеспечения, вне зависимости посредством каких технологий это программное обеспечение реализовывалось.

**Выбор веб - технологий**

На сегодняшний день в мире информационных технологий существует большое количество технологий, которые можно отнести к разряду веб - технологий. Выбор технологии для реализации информационной системы не простой процесс. Далеко не все специалисты, практикующие в этой области, могут точно и правильно подобрать технологии для реализации своего проекта. Для объективного выбора необходимо понимать специфику проекта, иметь солидный опыт работы на разных языках. Обычно подобный выбор, делается по каким либо субъективным причинам.

Разберём несколько языков, которые используются при реализации подобных проектов. Первое что нужно сделать – это определить критерии, на которые будем смотреть при выборе технологий.

- доступные инструменты разработки.
- существование готовых решений.
- наличие подробной документации.
- тренд развития технологии.
- кроссплатформенность.

- стоимость поддержки.

В технологиях можно обозначить три уровня абстракции:

**Чистый язык** – это в своём роде материал, из которого мы можем сделать всё что угодно. Здесь ограничителем могут быть только возможности самого языка программирования.

**Фреймворк** – это среда разработки с готовыми инструментами и правилами. Фреймворк ускоряет процесс разработки и способствует удобству работы с проектом, но с другой точки зрения он ставит определённые рамки при работе с проектом. На фреймворке реализуются проекты средней сложности.

**CMS**– это конструктор, готовое решение. Проект собирается по частям, скорее настраивают, чем программируют. Очень большое количество ограничений. Выйти за границы стандартной версии достаточно сложно.

**Характеристика популярных языков  их фреймворки и сайты реализованные на этих фреймворках:**

**PHP** – используется, как правило для достаточно средних и простых проектов. Последнее время имеет место анти-тренд. Но приобрёл значительные возможности после выхода последней версии (PHP 7). Фреймворки: Symfony, Laravel. Сайты: Вконтакте, ,Facebook, КиноПоиск

**Python**– один из самых современных языков из списка. Возможность достаточно качественной и быстрой разработки. Применяется при реализации больших и средних проектов. Фреймворк: Django. Сайты: Instagram, Reddit, Pinterest

**Ruby** – ещё один современный язык. Достаточно высокая скорость разработки. Применяется для реализации средних и простых проектов. Фреймворк: Ruby On Rails. Сайты: 500px, Airbnb, Groupon.

**Java** – очень долгая и дорогая разработка. Применяется при реализации очень крупных проектов со эксклюзивными требованиями. Фреймворк: Spring. Сайты: Ebay, Amazon, Alibaba.

Отталкиваясь от выше изложенного и проанализировав несколько опросов и рейтингов, оценивающих самые трендовые языки и технологии можно сделать определённые выводы.

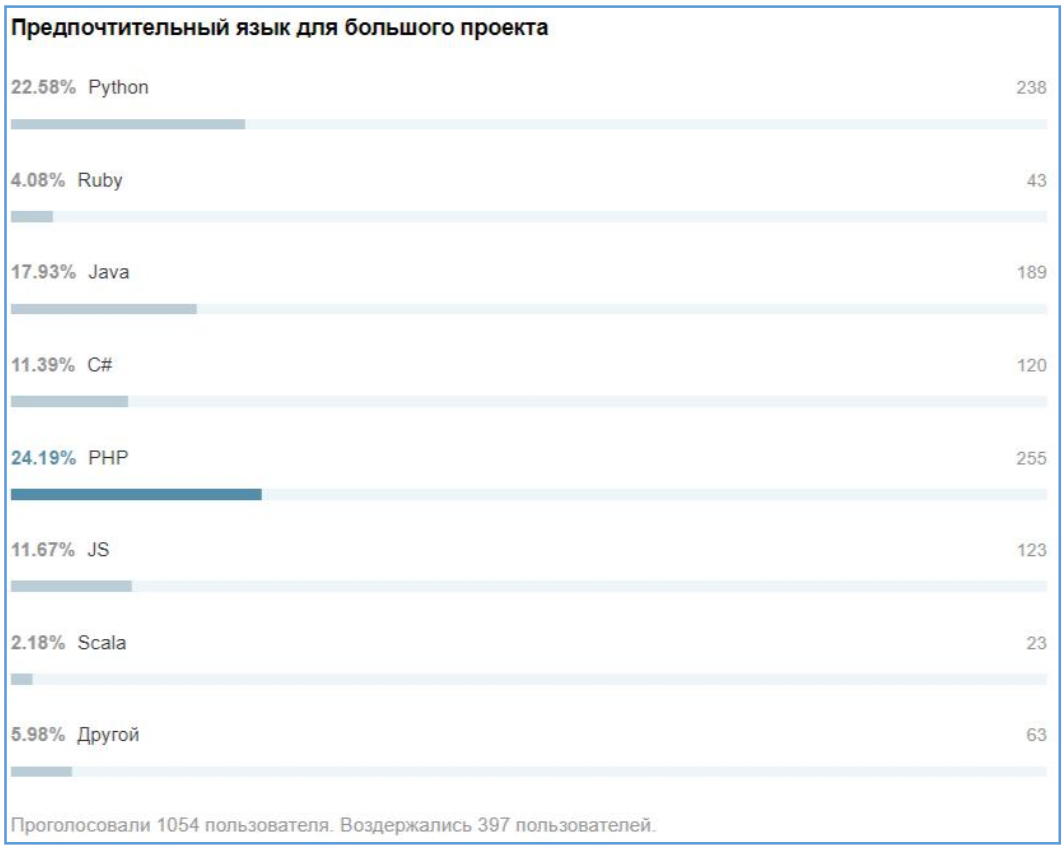



*Рисунок 6. Снимок экрана. Опрос на сайте habr.ru.*

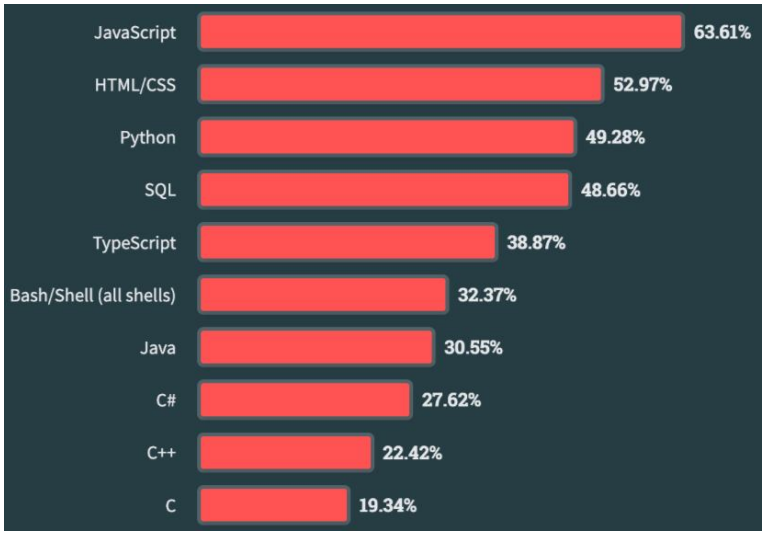



*Рисунок 7.Снимок экрана. Результаты опроса разработчиков 2023 года: последние тенденции в технологии и работе сообщества StackOverflow.*

На основе проведённой аналитики был сделан вывод - высокий рейтинг был отмечен у языка программирования Python. Исходя из того что этот язык простой в изучении, и в последние годы развитие этого языка набирает обороты выбор технологии для реализации информационной системы центра занятости было решено остановить именно на этом языке программирования и его фреймворке Django. Есть определённые обоснования такого выбора.

Создавать сайты на Python можно без применения сторонних фреймворков и библиотек. Создание сайта, используя только Python это долгий и сложный процесс. Для того чтобы упростить этот процесс были придуманы специальные фрэймворки или если сказать точнее готовые решения для упрощения процесса создания сайта. Фреймворк Django написан на основе Python с его помощью можно ускорить и упростить процесс создания как сложных веб-сайтов так и простых проектов. На самом деле Django не является монополистом и кроме него есть и другие фрэймворки обеспечивающие похожую функциональность. Не смотря на остальные фреймворки, Django уверенно ощущает себя на почётном месте лидера в сфере разработки веб-сайтов на Python. Преимущество Django перед другими похожими фреймворками в большом наборе уже готового функционала с помощью, которого можно сделать, например систему регистрации на сайте или форум.

В Django используется схемаMVC (model, view, controller) благодаря этой схеме есть возможность удобно разделить файлы проекта на несколько приведённых ниже категорий:

- Html-шаблоны.
- Модуль «Модели»(Models.py):всё необходимое для взаимодействия с базой данных.
- Модуль «Контроллеры» (Views.py): всё необходимое для взаимодействия моделей с Html-шаблонами.

На Django можно быстро создавать веб - сайты которые имеют быструю загрузку, а также отлично справляются с большими нагрузками на сервере. В каждом вновь создаваемом проекте по умолчанию подключается панель администратора, через эту панель можно управлять всем сайтом.

На Django написано огромное количество проектов например компания Google очень любит этот фреймворк и на его основе у них создано много проектов.

Для создания сайта на Django требуется знание языка Python поскольку весь синтаксис создаётся именно на языке программированияPython. Кроме знания языка Python необходимы навыки работы с языком разметки html и языком стилей css. В качестве альтернативного варианта вместо языка css можно использовать фреймворк Bootstrap.

**Язык программирования Python**

На данный моментPython – это один из самых популярных языков программирования. Создание веб – приложений считается одной из самых используемых сфер применения Python. Также язык получил очень широкое применение в сферах связанных с искусственным интеллектом. С помощью этого языка можно разрабатывать широкий спектр приложений: консольные, мобильные приложения, десктопные графические приложения. Python – интерпретируемый язык, то есть этот язык будет работать везде, где установлен интерпретатор. По факту это универсальный кроссплатформенный язык, он может работать на большинстве популярных платформ: Windows,MacOS, Linux.Для начала работы с языком Python необходимо установить его интерпретатор у себя на компьютере. Все необходимые файлы находятся на официальном сайте Python.

**Фреймворк Django**

Основная концепция фреймворка Django очень точно выражена в статье «Что такое Django?» размещённой на сайте metamit.com. В этой статье в частности говорится о том что «Django довольно популярен. Он используется на

многих сайтах, в том числе таких, как Pinterest, PBS, Instagram, BitBucket, Washington Times, Mozilla и многих других.

Фреймворк является бесплатным. Он развивается как open source, его исходный код открыт, его можно найти репозитории на githube.

На Django можно создавать широкий диапазон веб -приложений: от небольших персональных сайтов до самых высоконагруженных сложных веб - сервисов.

Django по умолчанию предлагает готовую функциональность для ряда распространенных задач, например, систему аутентификации, генерацию карт сайта и т.д., благодаря чему нам можно не изобретать велосипед и достаточно взять уже готовые компоненты.

В Django большое внимание уделяется безопасности, благодаря чему фреймворк помогает разработчикам избежать многих распространенных проблем в системе безопасности, например, sql-инъекций.

Фреймворк Django реализует архитектурный паттерн **Model-View-Template** или сокращенно **MVT**, который по факту является модификацией распростаненного в веб - программировании паттерна MVC (Model-View-Controller).

Схематично мы можем представить архитектуру MVT в Django следующим образом:

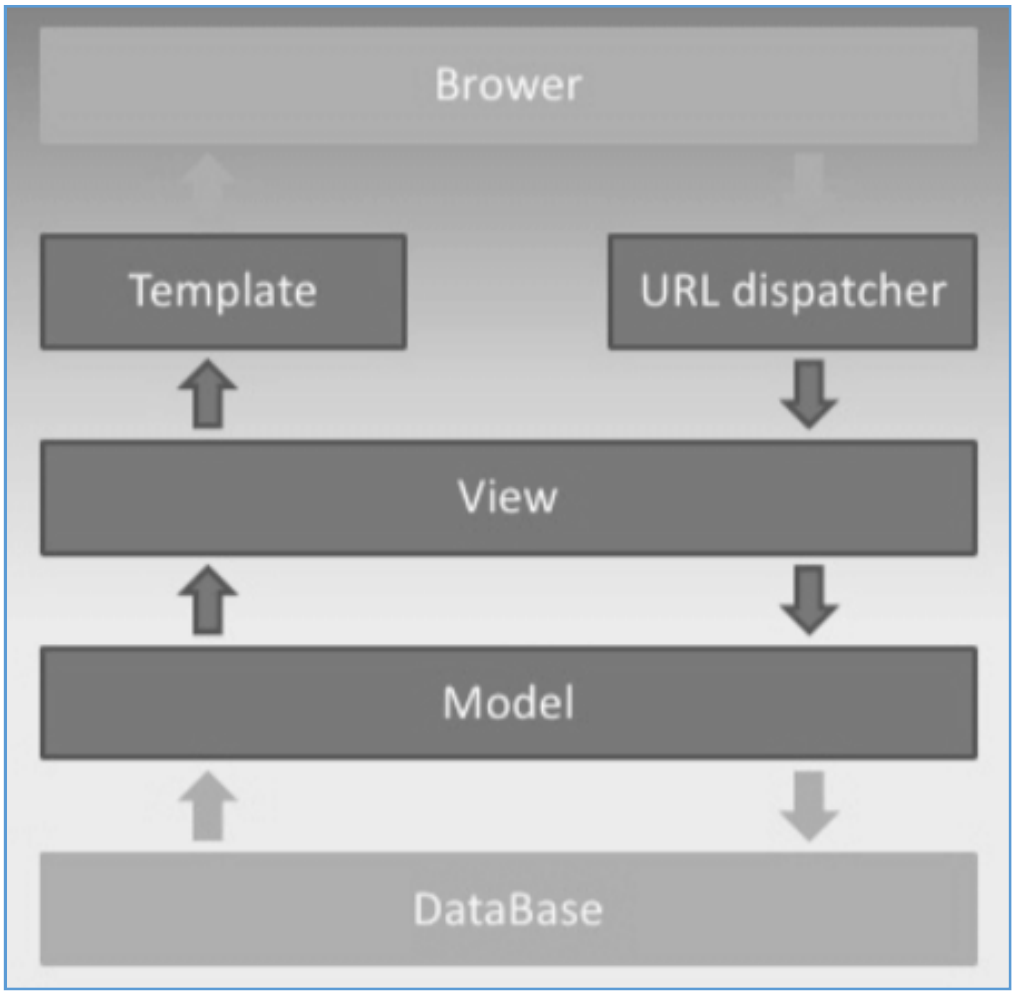



*Рисунок 8. Схема архитектуры MVT в Django.*

Основные элементы паттерна:

**URL dispatcher**: при получении запроса на основании запрошенного адреса URL определяет, какой ресурс должен обрабатывать данный запрос.

**View**: получает запрос, обрабатывает его и отправляет в ответ пользователю некоторый ответ. Если для обработки запроса необходимо обращение к модели и базе данных, то View взаимодействует с ними. Для создания ответа может применять Template или шаблоны. В архитектуре MVC этому компоненту соответствуют контроллеры (но не представления).

**Model**: описывает данные, используемые в приложении. Отдельные классы, как правило, соответствуют таблицам в базе данных.

**Template**: представляет логику представления в виде сгенерированной разметки html. В MVC этому компоненту соответствует View, то есть представления.

Когда к приложению приходит запрос, то URL dispatcher определяет, с каким ресурсом сопоставляется данный запрос и передает этот запрос выбранному ресурсу. Ресурс фактически представляет функцию или View, который получает запрос и определенным образом обрабатывает его. В процессе обработки View может обращаться к моделям и базе данных, получать из нее данные, или, наоборот, сохранять в нее данные. Результат обработки запроса отправляется обратно, и этот результат пользователь видит в своем браузере. Как правило, результат обработки запроса представляет сгенерированный html-код, для генерации которого применяются шаблоны (Template)» [4].

Для того чтобы начать работать с Django интерпретатор Python уже должен быть установлен на компьютере.

В качестве текстового редактора для написания программного кода можно использовать любую доступную программу. Например SublimeText, Visual Studio или PyCharm.

**Таблицы стилей CSS.**

Отталкиваясь от материалов статьи «Введение в стили» опубликованной на сайте «METANIT.COM» делаем вывод что «Любой html-документ, сколько бы он элементов не содержал, будет по сути "мертвым" без использования стилей. Стили или лучше сказать каскадные таблицы стилей (**Cascading Style Sheets**) или попросту CSS определяют представление документа, его внешний вид.

Стиль в CSS представляет правило, которое указывает веб - браузеру, как надо форматировать элемент. Форматирование может включать установку цвета фона элемента, установку цвета и типа шрифта и так далее.

Определение стиля состоит из двух частей: **селектор**, который указывает на элемент, и **блок объявления стиля** - набор команд, которые устанавливают правила форматирования.

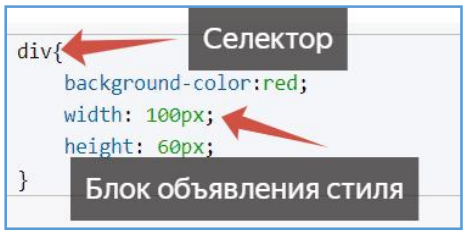



*Рисунок 9. Формулировка стиля CSS.*

Для определения стилей могут использоваться различные способы. Первый способ состоит в непосредственном встраивании стилей непосредственно в html - элемент с помощью специального атрибута style.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Стили</title>
    </head>
    <body>
        <h2 style="color:blue;">Стили</h2>
        <div style="width: 100px; height: 100px; background-color: red;"></div>
    </body>
</html>
```

*Рисунок 10. Использование атрибута style.*

Второй способ заключается в применении элемента style в документе html. В этом случае элемент style объявляет браузеру, что данные, которые находятся внутри этого блока, являются таблицей стилей, а не Html.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Стили</title>
        <style>
        h2{
            color:blue;
        }
        div{
            width: 100px;
            height: 100px;
            background-color: red;
        }
        </style>
```

*Рисунок 11. Использование элемента style.*

Существует ещё один способ использования стилей это использование элемента link. В этом элементе прописывается ссылка до файла стилей.» [5].

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="styles.css"/>
```

*Рисунок 12. Использование элемента link.*

**Фреймворк Bootstrap.**

Bootstrap – это бесплатный набор инструментов для производства сайтов а также веб - приложений. Этот фреймворк состоит из шаблонов Html и CSS. Этот фреймворк используется для того же что и CSS, то есть для оформления страниц сайта. О возможных способах подключения Bootstrap к сайту рассказано в статье «Подключение фреймворка Bootstrap к сайту» размещённой на сайте «ИТШЕФ». В частности на этом ресурсе сказано, что «Подключить Bootstrap к сайту можно разными способами, например:

- с помощью CDN (в этом варианте помещать файлы непосредственно в проект не нужно);

- локально с настройками по умолчанию (для этого необходимо скачать дефолтную готовую сборку Bootstrap с GitHub, извлечь из неё необходимые файлы и подключить их к HTML странице);
- локально со своими настройками (для этого необходимо скачать исходные коды и программу для сборки frontend).» [6].

**СУБД PostgreSQL.**

PostgreSQL – это реляционная база данных с открытым исходным кодом на сегодняшний день она является одной из наиболее известных СУБД. Данная СУБД входит в пятёрку лучших СУБД по ряду общепризнанных мировых рейтингов. Кроме того на основе PostgreSQL была создана отечественная СУБД PostgresPro, которая входит в реестр российского программного обеспечения. В связи с проведением в данный момент программы импортозамещения данная СУБД очень плотно входит в нашу жизнь.

**ORM.**

ORM это объектно-реляционное связывание (Object Relational Mapping) – комплекс программ, который позволяет работать с базой данных, так как если бы они были объектами языка программирования, в нашем случае Python. Иначе ORM это набор классов, который добавляет ещё один уровень абстракции к таблицам.

Посредством ORM и PostgreSQL в информационной системе для службы занятости будет реализовано взаимодействие между пользовательским интерфейсом и базой данных.

Таким образом, оптимальным набором технологий для разработки информационной системы для службы занятости является:

- язык программирования Python;
- фреймворк Django;
- язык разметки Html;
- фреймворк Bootstrap;

- СУБД PostgreSQL;
- технология ORM.

## 1.3 Техническое задание на разработку информационной системы для службы занятости

Техническое задание, составленное на основе ГОСТ 34.602–89 [**Ошибка! Источник ссылки не найден.**].

***Общие сведения.***

*Название организации–заказчика.*

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Уральский государственный педагогический университет».

*Название продукта разработки.*

Информационная система для службы занятости.

*Назначение продукта.*

Автоматизация процесса доступа к информации о резюме и вакансиях службы занятости.

*Плановые сроки начала и окончания работ.*

В соответствии с планом выполнения ВКР (01.09.2023 – 15.02.2024).

***Характеристика области применения продукта.***

*Процессы и структуры, в которых предполагается использование продукта разработки.*

Информационная система будет использована в сфере информирования пользователей об актуальных вакансиях и резюме добавленных другими пользователями.

*Характеристика персонала (количество, квалификация, степень готовности).*

Разработчик должен знать фреймворк Django и язык программирования Pythonна уровне не ниже базового.

Пользователь должен иметь уровень владения ПК не ниже базового. Умение пользоваться интернет - браузером. Знание общих правил использования интернет – ресурсов.

***Требования к продукту разработки.***

*Требования к продукту в целом.*

Информационная система для размещения информации о вакансиях и резюме для службы занятости.

*Аппаратные требования.*

Персональный компьютер, смартфон или любое другое устройство, на котором можно запустить интернет-браузер и имеющее выход в интернет.

*Указание системного программного обеспечения (операционные системы, браузеры, программные платформы и т.п.).*

Операционные системы: iOS(версии 9 и выше), Android (версии 4.1 и выше), MacOS, Windows 7 и выше.

Браузеры: любой браузер способный запуститься на целевой операционной системе.

*Указание программного обеспечения, используемого для реализации.*

Visual Studio Code 1.85.1

*Форматы входных и выходных данных*

Входные данные: данные введённые пользователем во время заполнения соответствующих форм на сайте. Текстовые данные, загрузка растровых изображений (.jpg, .png).

Выходные данные:html – страницы, сформированные в соответствии с запросом пользователя.

*Источники данных и порядок их ввода в систему (программу), порядок вывода, хранения.*

Подготовка источников данных не является частью разработанной информационной системы и подготавливается самим пользователем. Данные вводятся пользователем в соответствующие формы на сайте, после чего введённые данные сохраняются в базе данных. При обращении пользователя к этим данным они выводятся в специально сформированной html-странице.

*Порядок взаимодействия с другими системами, возможности обмена информацией.*

Взаимодействие с другими системами не предусмотрено.

*Меры защиты информации.*

Доступ к редактированию содержимого информационной системы предоставляется только пользователям являющимися авторами этого содержимого. Также защита информации предусмотрена возможностями самого фреймворка Django, посредством стандартных валидаторов.

***Требования к пользовательскому интерфейсу.***

*Общая характеристика пользовательского интерфейса.*

Интерфейс определяется встроенным функционалом фреймворка Django и фреймворка Bootstrap.

*Размещение информации на экране, дизайн экрана.*

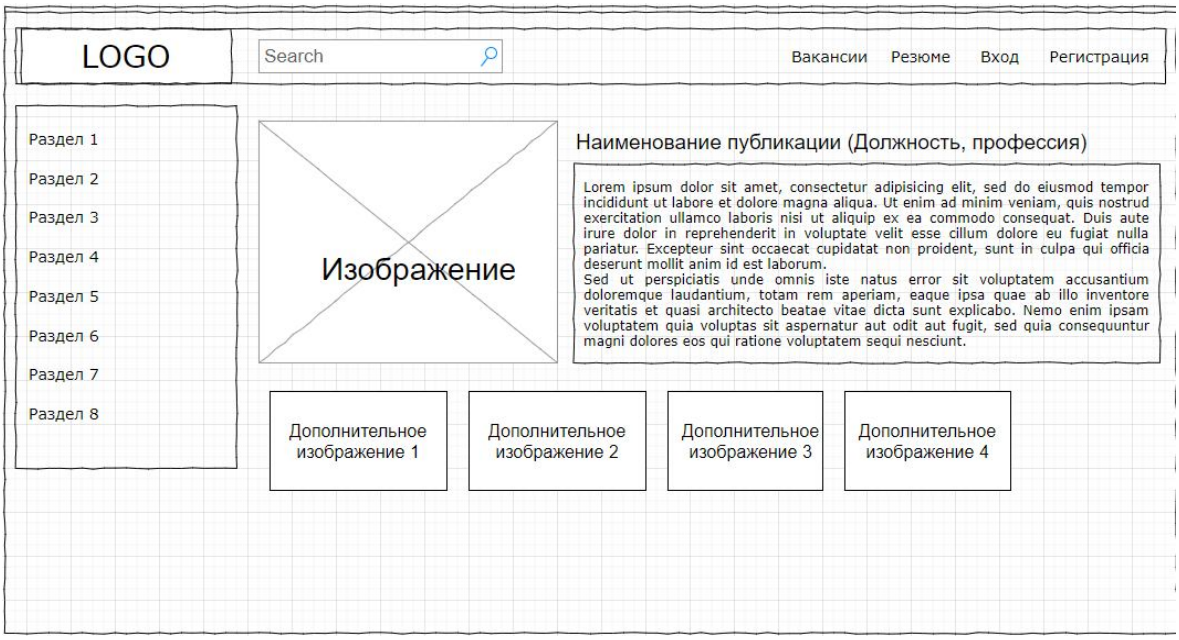


*Рисунок 13. Вид главной страницы.*

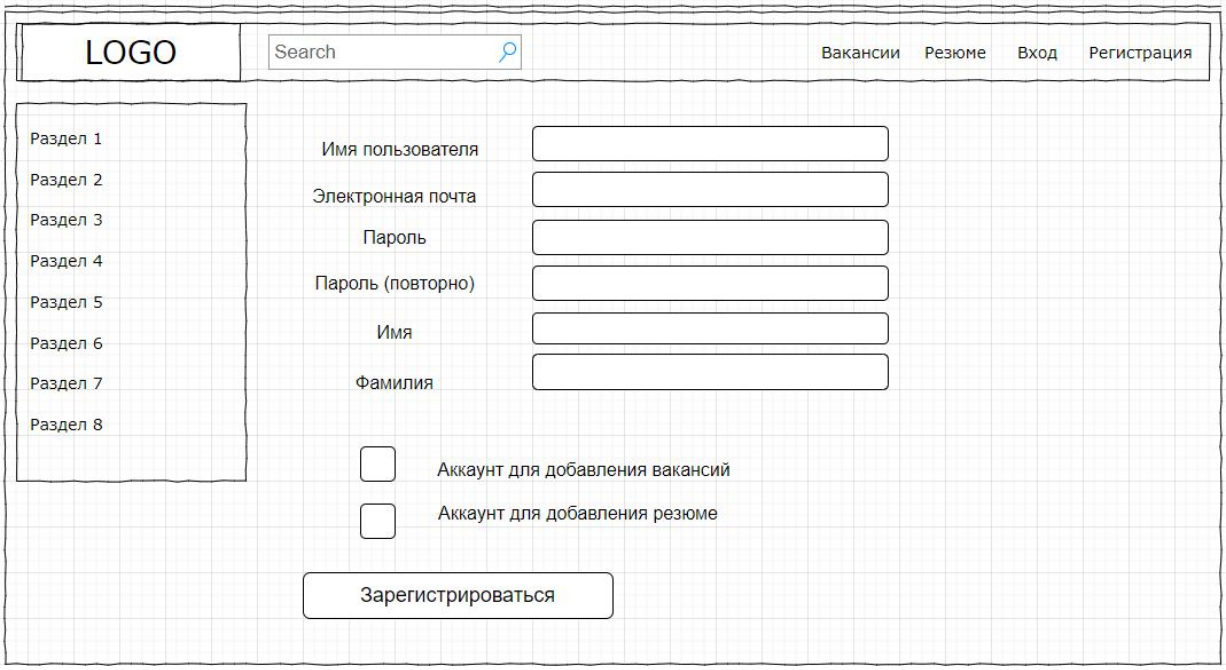



*Рисунок 14. Форма регистрации пользователе.*

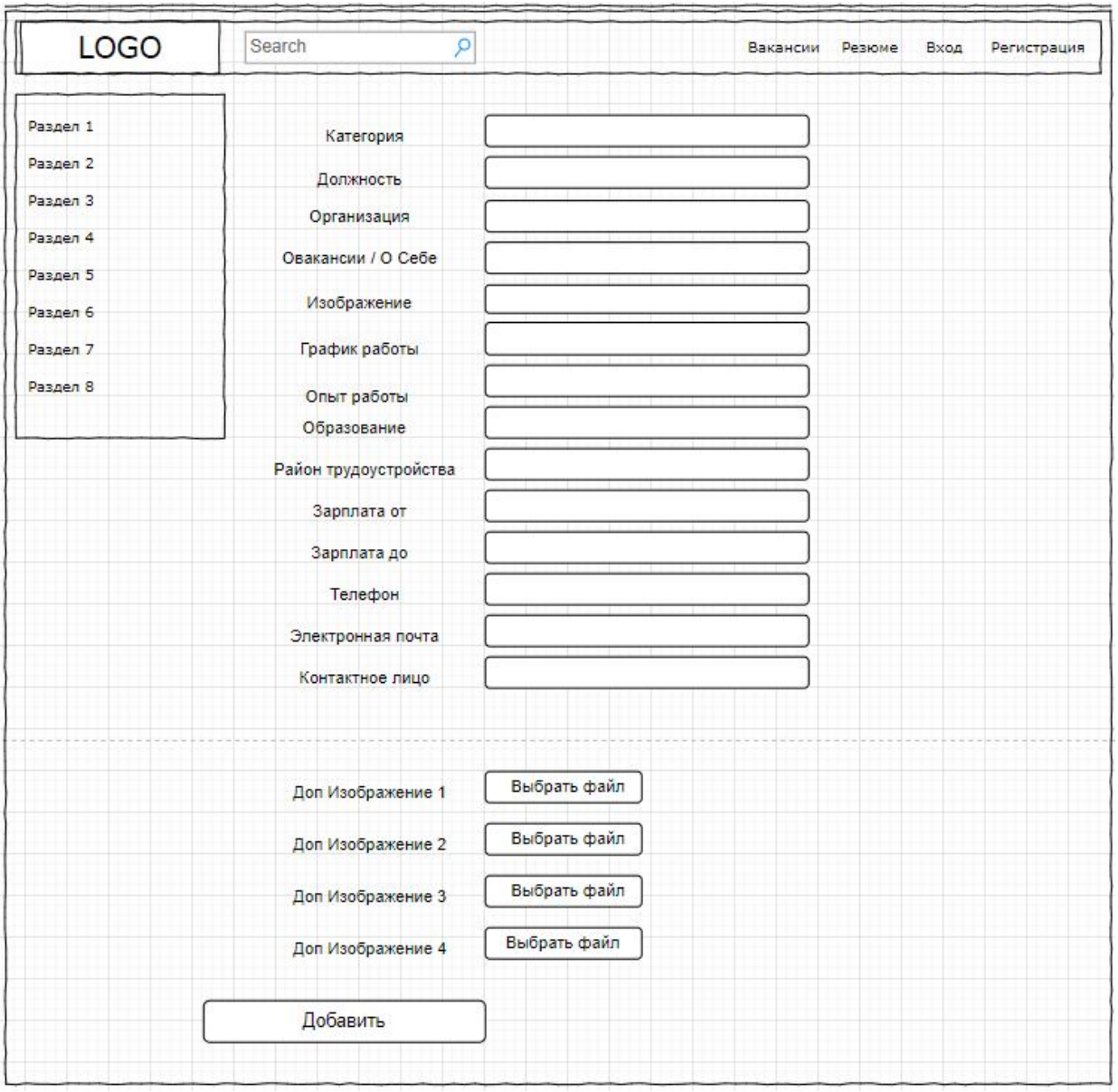



*Рисунок 15. Форма добавления публикации на сайт.*

*Особенности ввода информации пользователем, представление выходных данных.*

Информация вводится с клавиатуры, либо выбираются значения из списка.

*Требования к документированию.*

*Перечень сопроводительной документации.*

Техническое задание – главный документ, сопровождающий информационную систему для центра занятости.

*Порядок сдачи–приемки продукта.*

В соответствии с планом выполнения ВКР. Продукт считается принятым при его положительной оценке экспертами – преподавателями вуза.

# Глава 2. Реализация информационной системы для службы занятости

## 2.1 Модельные представления объекта разработки

Сегодня для облегчения реализации моделей бизнес-процессов в компаниях используется ряд методов. Использование моделей не только повышает опыт и практические навыки, облегчает понимание моделей, созданных другими сотрудниками, но и обеспечивают общепринятый подход к описанию.

**Структурный (функциональный) подход**

Суть структурного подхода к построению информационной системы заключается в разделении на автоматизированные процессы. Основная идея заключается в том, чтобы разделить всю систему на ряд функциональных подсистем, разделить подсистемы на подфункции, а подфункции - на задачи и так далее. В рамках функции проектирования и структурного анализа обычно используется следующая нотация:

(IDEF0) используется для определения требований к созданию системы, реализующей выбранную функцию. Учитывая существующие IDEF0, их, безусловно, можно использовать их для анализа функций, выполняемых системой. Таким образом, модель в нотации IDEF0 напоминает набор иерархически связанных диаграмм (рис. 17). На вершине этой древовидной структуры находится самое общее описание системы. Далее по ходу описания система декомпозируется более на крупные части ( функциональная декомпозиция).

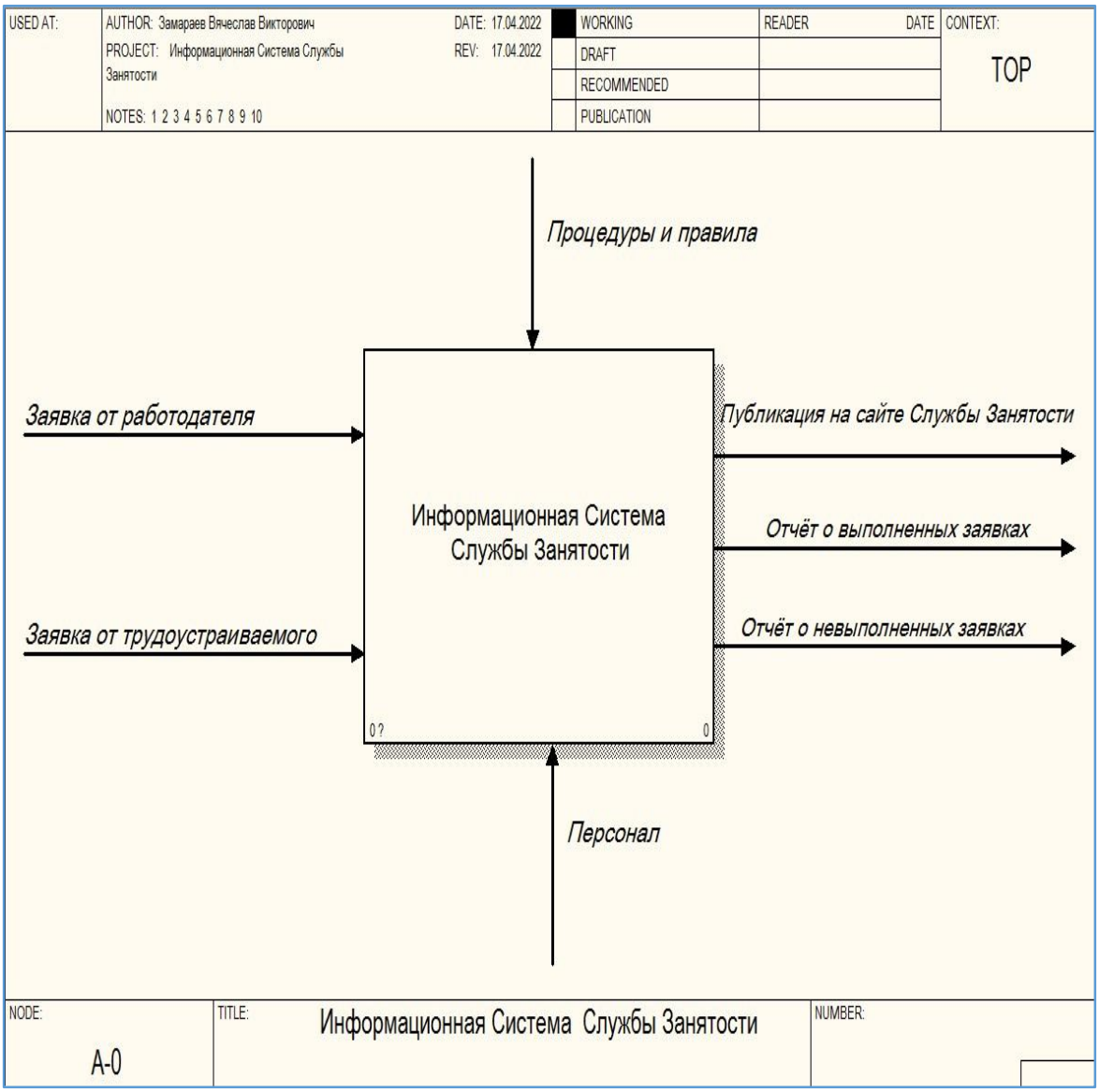



*Рисунок 16. Модель в нотации IDEF0*

DFD (Data Flow Diagrams) диаграммы потоков данных. Этот тип диаграмм используется в качестве дополнения к моделям бизнес-процессов, в нотации IDEF0.

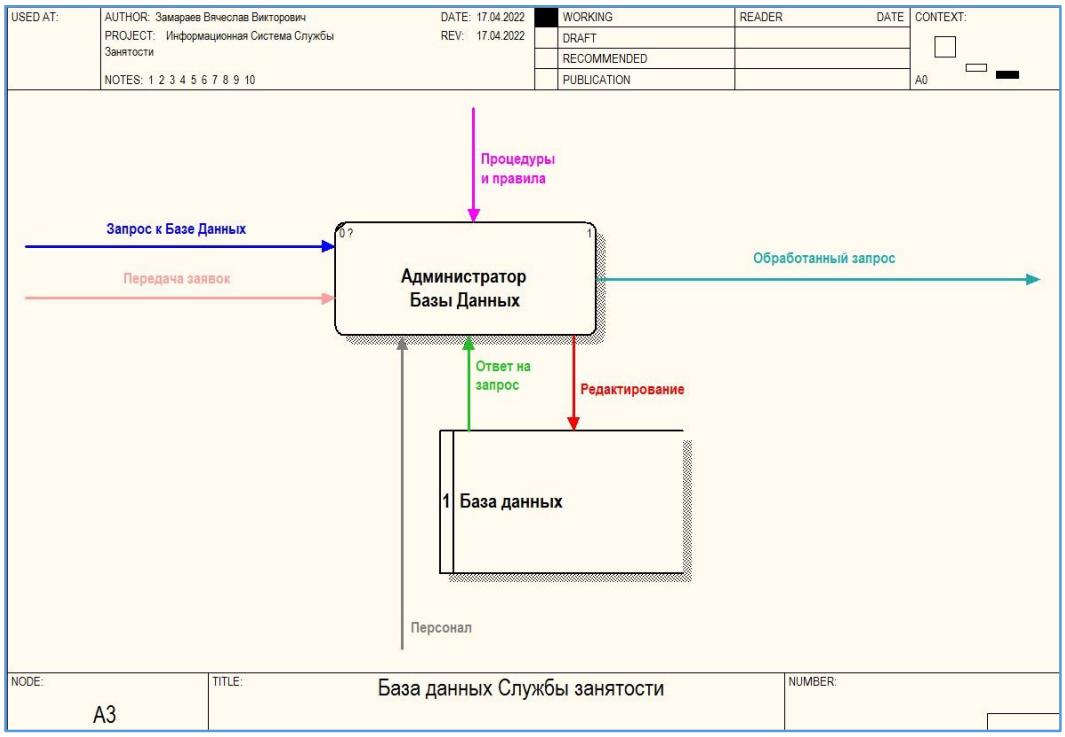



*Рисунок 17. Модель в нотации DFD*

Общий вывод: использование универсальных языков графического моделирования IDEF0 и DFD позволяет обеспечить логическую последовательность и максимальную полноту описания, необходимые для получения точных и непротиворечивых результатов на этапе анализа.

Наиболее существенное различие между различными типами структурного анализа заключается в их функциональности.

Модели IDEF0 проще всего использовать при формировании функциональных моделей. Модели IDEF0 чётко представляют функциональную структуру объекта (выполняемые действия, связи между действиями). При этом чётко прослеживается логика и взаимодействие между процессами организации. Основное преимущество этой нотации заключается в том, что строго определённая структура предоставляет исчерпывающую информацию о каждой такой операции. Через эту структуру можно выявить все недостатки, относящиеся как к самому процессу, так и к средствам его осуществления. Поэтому DFD-диаграммы обычно используются в качестве дополнения к моделям бизнес-процессов, реализованным в IDEF0.

IDEF3 используется для сбора данных, необходимых для проведения системного анализа с точки зрения не соответствия/согласованности процессов во времени.

Нельзя говорить о преимуществах и недостатках той или иной нотаций. Существуют ситуации, когда аналитика IDEF0 не может обнаружить ошибки в деятельности организации с точки зрения производственного или технологического процесса, но, к сожалению это не гарантирует отсутствия ошибок. Поэтому следующим этапом анализа должен стать переход к изучению информационных потоков с помощью DFD, а затем соединение этих пространств с помощью конечной нотации - IDEF3.

В данной работе применяется структурный подход к моделированию. Этот подход даёт возможность детально проанализировать бизнес – процессы и выявить узкие места. Использование языков графического моделирования IDEF0, DFD, создаёт полноту описания и логическую последовательность, необходимые для достижения согласованных и точных результатов при моделировании предметной области в контексте «как есть» и «как должно быть»;

Большинство людей, участвующих в реализации любого проекта, связанного с созданием новой или развитием существующей информационной системы, согласятся с тезисом о том, что заказчикам нужны информационные системы, которые позволят им повысить эффективность работы своей организации. К сожалению, разработчики информационных систем и заказчики, как и прежде, общаются на разных языках. У них нет общего ответа на вопрос, что значит повысить эффективность организации.

Для того чтобы разработчики и заказчики информационных систем понимали друг друга, необходимо, чтобы разработчики систем сменили направление деятельности с решения вопросов технического планирования, связанных с построением и совершенствованием информационных систем, на решение комплексных проблем, связанных с повышением функциональной эффективно-

сти организации. информационной системы на решение комплексных проблем по увеличению эффективности функционирования организации:

-изучение бизнес-структур, бизнес-правил, информационных потоков;

-обнаружение так называемых - "узких" мест, проблем, негативно влияющих на эффективность организации;

-внедрение и разработка процедур и реструктуризация бизнес-процессов для устранения существующих проблемных ситуаций и улучшения бизнес-структуры предприятия;

разработка конкретных проектов информационных систем, реализация этих проектов и их дальнейшее сопровождение.

В рамках данной методологии на повышение эффективности работы разработчиков информационных систем нацелен специальный инструментарий, используемый для моделирования организации и реинжиниринга бизнес-процессов. Примером данного семейства инструментов является CASE-средства, предназначенные для функционального моделирования бизнес-процессов.

***Построение модели центра занятости «как есть»***

Изучение объектов проектирования считается необходимой частью любого проекта, как при создании, так и при разработке информационных систем. Создание функциональной модели «как есть» даёт чёткое представление о том, какие процессы происходят в организации, и какие информационные объекты используются для реализации процессов и отдельных задач. Функциональная модель «как есть» считается отправной точкой для анализа потребностей предприятия.

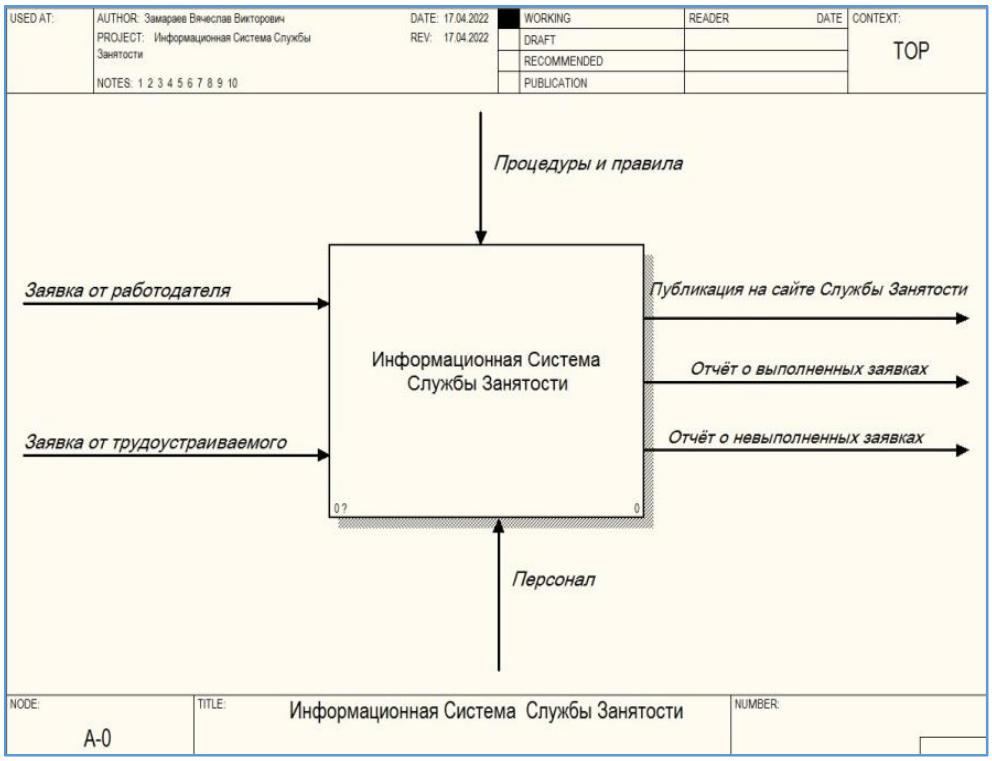



*Рисунок 18. Контекстная диаграмма службы занятости.*

***Контекстная диаграмма*** "фиксирует" границы моделируемой бизнес-системы, а также определяет, как моделируемая система взаимодействует со своим окружением. Это достигается путём описания дуг, связанных с блоками, представляющим ключевые бизнес– функции. На верхнем уровне контекстная диаграмма представлена как «чёрный ящик», в котором происходит некоторая деятельность, преобразующая входы в выходы.

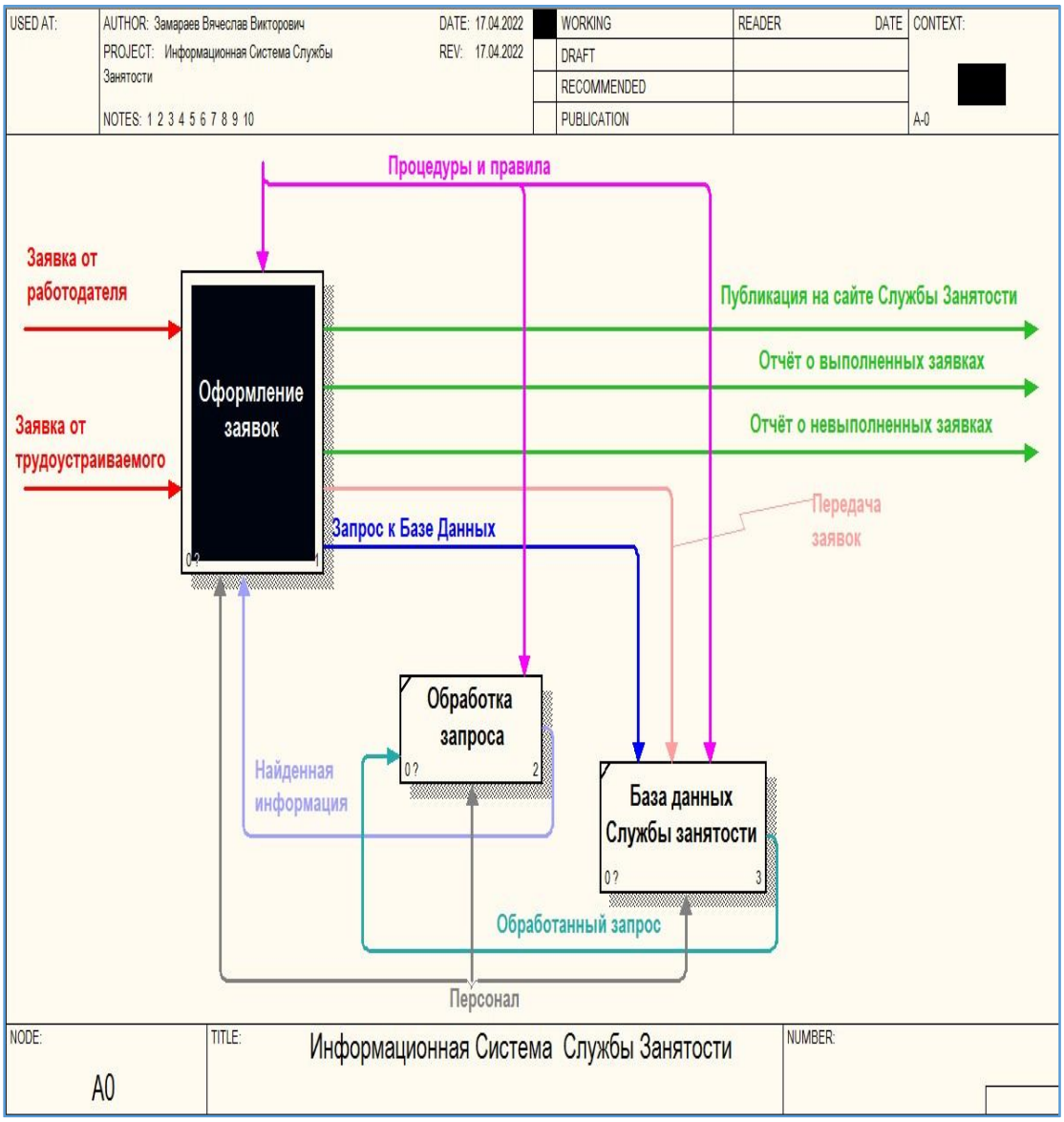



*Рисунок 19. Декомпозиция контекстной диаграммы службы занятости.*

***Декомпозиция*** – это процесс «углубления» в рассматриваемую функцию, путём её разделения на более мелкие функции.В данном случае целесообразно представить функцию верхнего уровня обобщённо, а после того как она будет декомпозирована её уже можно будет называть процессом.

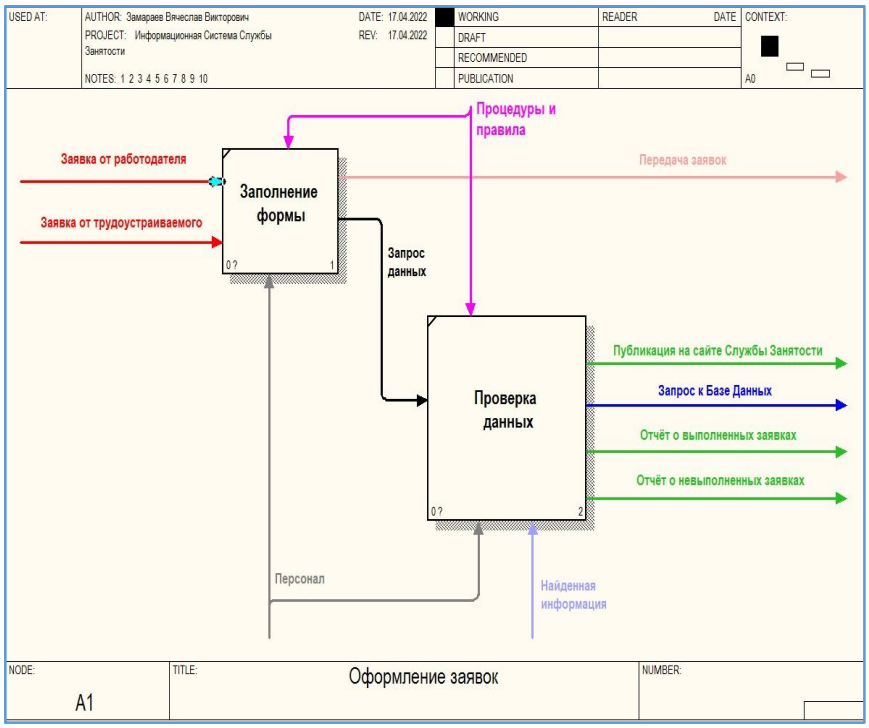



*Рисунок. 20. Декомпозиция модуля оформления заявок.*

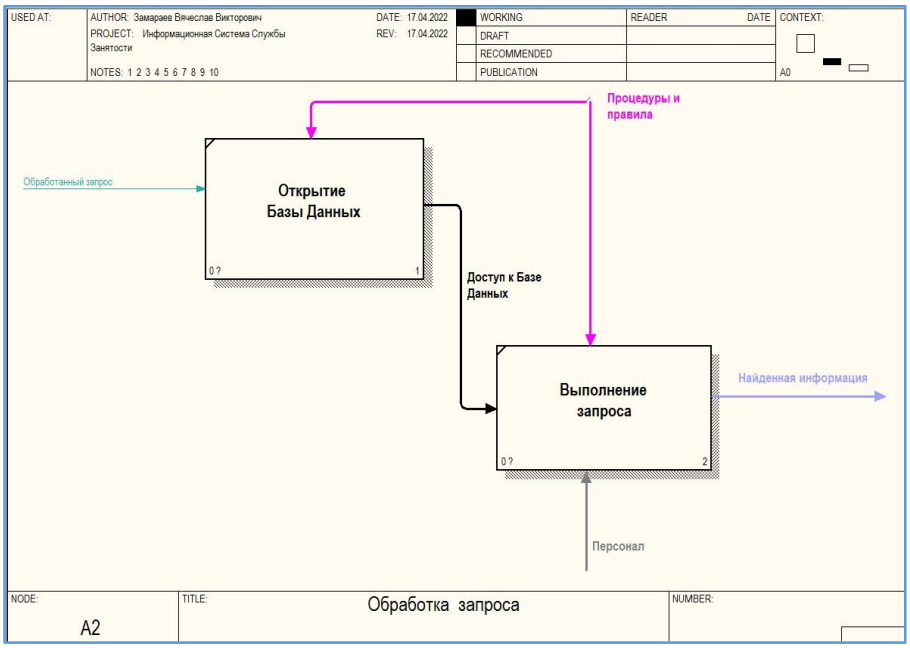



*Рисунок 21. Декомпозиция модуля – обработка запроса.*

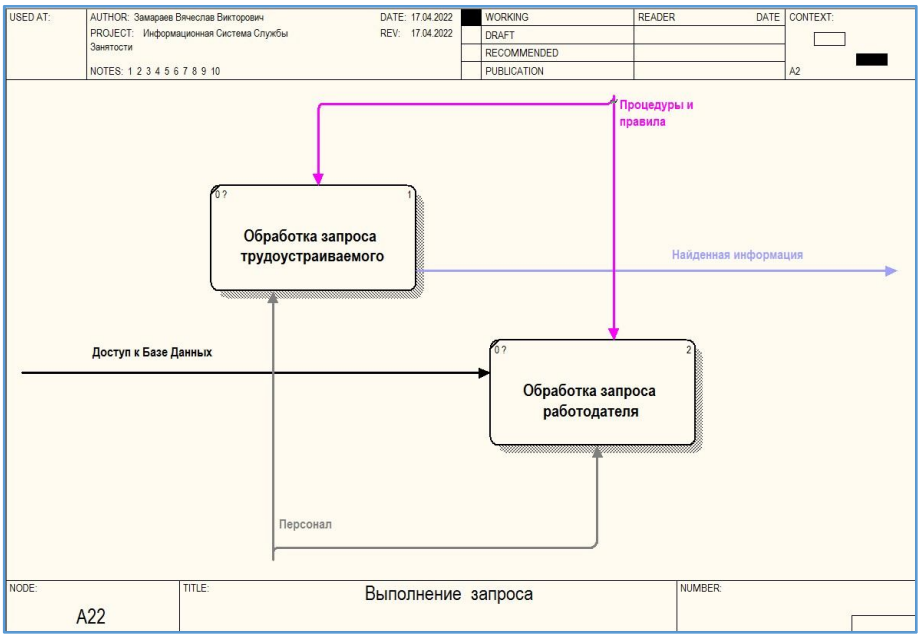



*Рисунок 22. Декомпозиция в нотации DFD – выполнение запроса.*

***DFD*** – это стандарт моделирования, который представляет систему как сеть действий, связанных объектами, которые взаимодействуют с результатами этих действий.

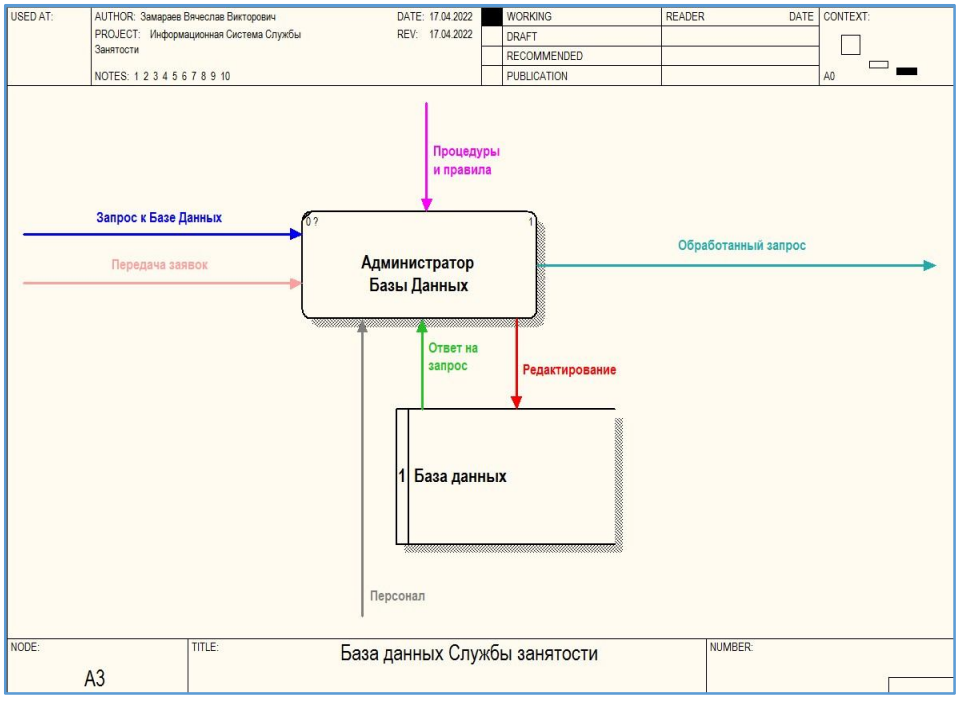



*Рисунок 23. Декомпозиция в нотации DFD – база данных службы занятости.*

## 2.2 Реализация информационной системы службы занятости

Разрабатываемая информационная система позволит зарегистрированным пользователям создавать публикации о вакансиях и резюме. Публикации будут распределяться по категориям (разделам), структура категорий будет иметь двухуровневую иерархию: на первом уровне категории общего плана («Здравоохранение»,«ИТ»,«Образование»,«Промышленность»,«Силовые структуры») на втором уровне – более конкретные, например для категории «Образование» будут очевидны такие подкатегории как: «Высшее», «Профессиональное», «Среднее», «Дошкольное», «Дополнительное». С учётом того что сайт будет иметь возможность расширятся, никаких ограничений для количества категорий и их подкатегорий не будет.

Для того чтобы выводить список публикаций будет использована пагинация, так как публикаций может быть очень много, и страница, содержащая все объявления, будет очень объёмной. Будет предусмотрена возможность поиска необходимого объявления по введённому пользователем слову.

На сайте будет возможность добавления произвольного количества комментариев под каждой публикацией. Возможность добавить комментарий будет предоставлена любому пользователю, в том числе и гостю.

В своей публикации пользователь может поместить главную графическую иллюстрацию, которая будет отображаться в списке публикаций, и в составе сведений о публикации. А также возможно добавление произвольного количества дополнительных иллюстраций, которые будут доступны лишь на странице сведений о публикации. Как основная, так и дополнительная иллюстрации к публикации не будут являться обязательными.

Сайт разрабатываемой информационной системы будет включать в себя следующие страницы:

- Главная страница – показывающая десять последних опубликованных публикаций без разбиения их на рубрики;
- Страница выводящая список публикаций – отображает, посредством пагинации, публикации из определённой рубрики. Кроме этого данная страница будет содержать форму для поиска публикаций по введённому слову;
- Страница, выводящая сведения о выбранной публикации – покажет дополнительно все оставленные для данной публикации комментарии и форму для добавления нового комментария;
- Страницы для регистрации и активации нового пользователя;
- Страницы входа и выхода;
- Страница профиля зарегистрированного пользователя – покажет список публикаций, оставленных текущим пользователем.
- Страницы добавления, правки, удаления объявлений;
- Страницы изменения пароля, правки и удаления профиля пользователя;

Любой пользователь, который заходит на главную страницу сайта информационной системы для службы занятости имеет возможность видеть последние 10 публикаций размещённых на сайте.

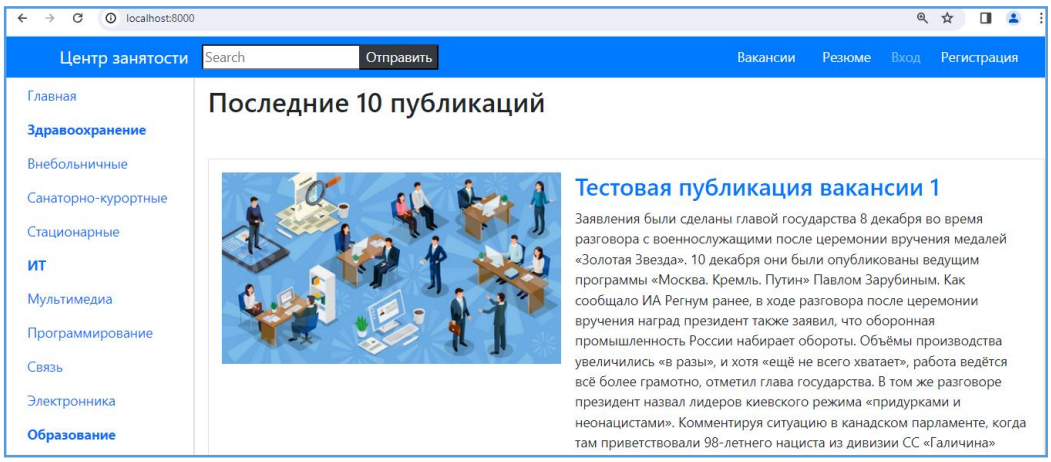



*Рисунок 24. Главная страница для незарегистрированного пользователя.*

Кроме того в верхнем меню доступно несколько пунктов предоставляющие базовые возможности для пользователей сайта. Незарегистрированный пользователь имеет возможность просматривать все разделы сайта. Также он может добавлять к любым публикациям комментарии, единственным ограничением при добавлении комментария будет необходимость ввода капчи.

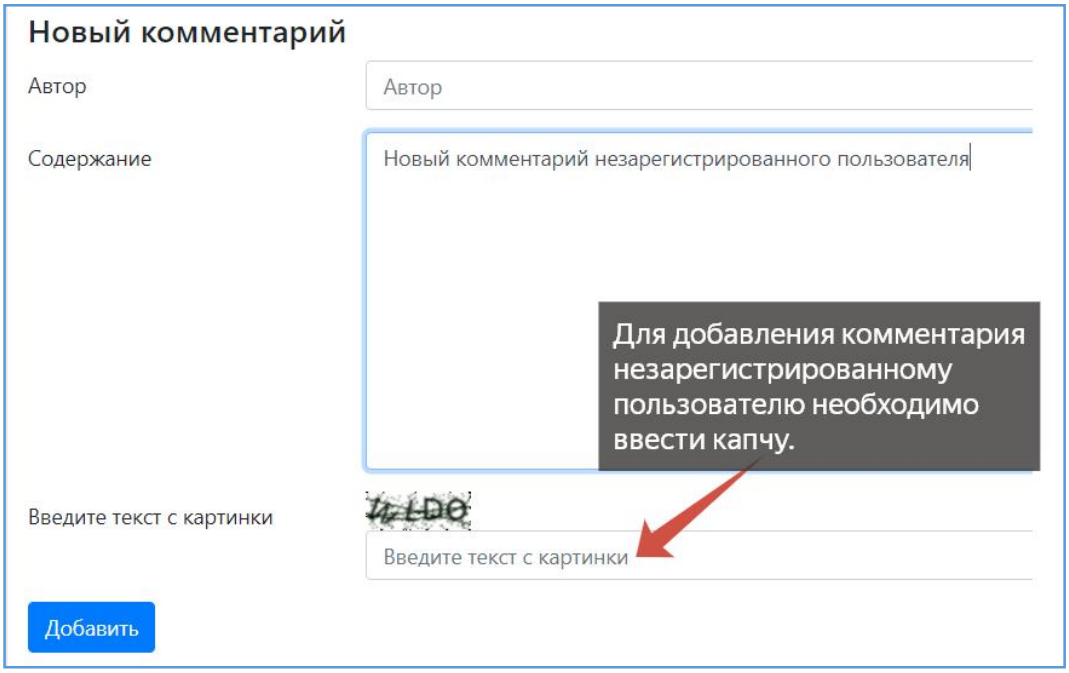



*Рисунок 25. Добавление комментария для незарегистрированного пользователя.*

Также в верхнем меню присутствует возможность для регистрации на сайте, а также для входа на сайт – в том случае если у пользователя уже есть учётная запись для входа на сайт.

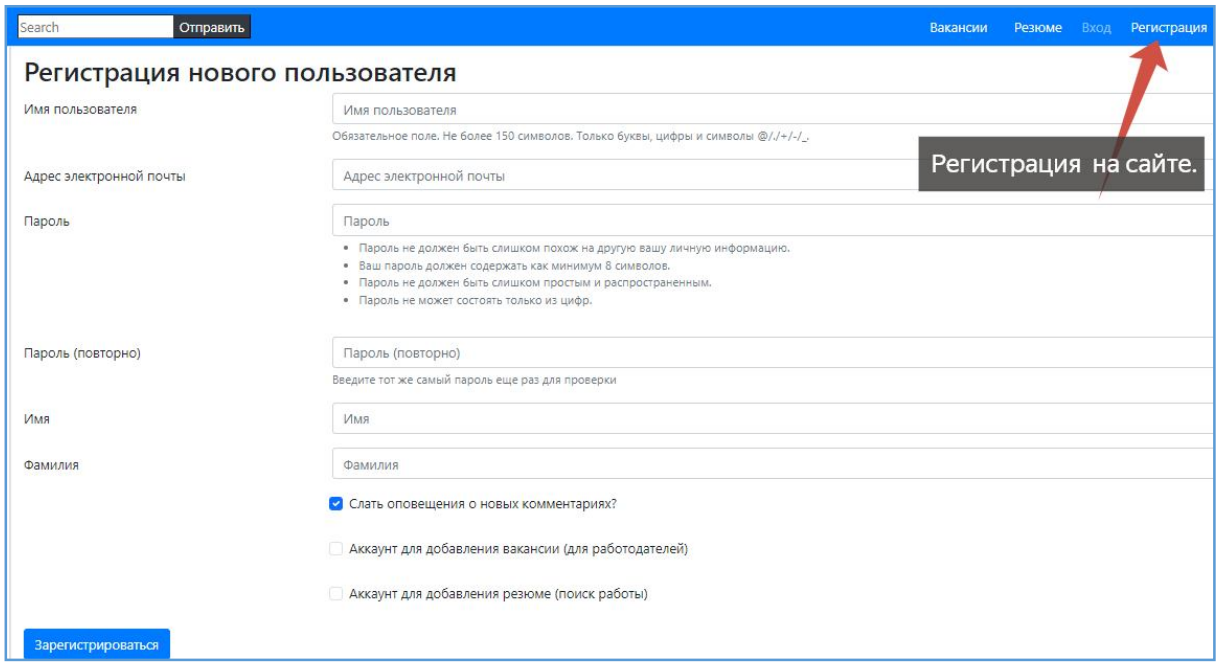



*Рисунок 26. Форма регистрации на сайте.*

Для регистрации на сайте необходимо ввести: имя пользователя, email, пароль, имя, фамилию. Кроме этого добавлена возможность отметить необходимо ли высылать оповещения о новых комментариях на электронную почту, а также необходимо выбрать тип аккаунта в зависимости от того какой тип публикаций пользователь будет добавлять на сайт. Предусмотрено два варианта: аккаунт для добавления вакансий (как правило, это работодатели) и аккаунт для добавления резюме (для пользователей ищущих работу).

После нажатия на кнопку «Зарегистрироваться» происходит процесс регистрации, и на указанный при регистрации адрес электронной почты будет отправлено письмо с ссылкой, перейдя по которой пользователь активирует свою учётную запись. Только после активации своей учётной записи пользователь сможет зайти под своими учётными данными на сайт.

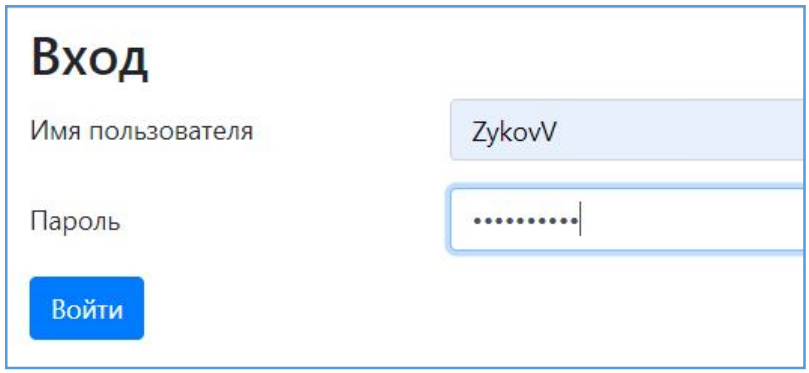



*Рисунок 27. Форма входа на сайт.*

Если пароль или логин будут введены не правильно или учётная запись ещё не активирована, то сработает встроенный в Django валидатор, будет выведено сообщение об ошибке.

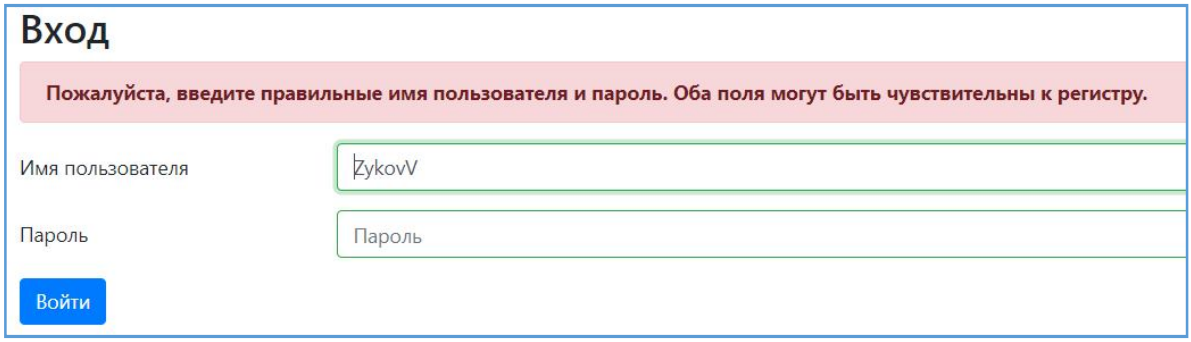



*Рисунок 28. Сообщение об ошибке при неправильном вводе логина или пароля.*

После входа пользователя на сайт у него появляются дополнительные возможности. В верхнем меню вместо пункта «Вход» появляется пункт меню «Профиль».

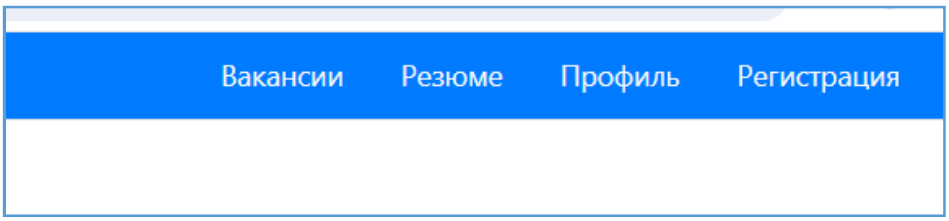



*Рисунок 29. Изменение пункта меню после входа на сайт.*

Пункт меню «Профиль» содержит в себе инструменты, управляющие учётной записью пользователя.

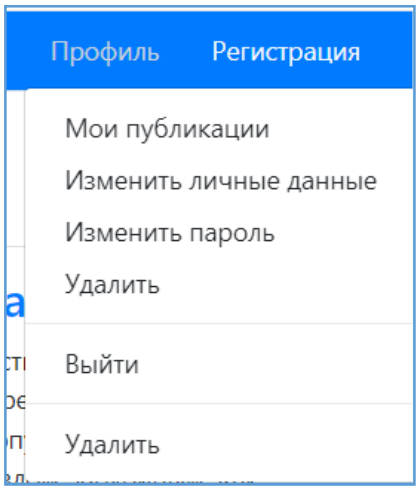



*Рисунок 30. Меню «Профиль».*

Пункт меню «Мои публикации» содержит все публикации текущего пользователя. И соответственно инструменты управления этими публикациями. Можно добавить новую публикацию.

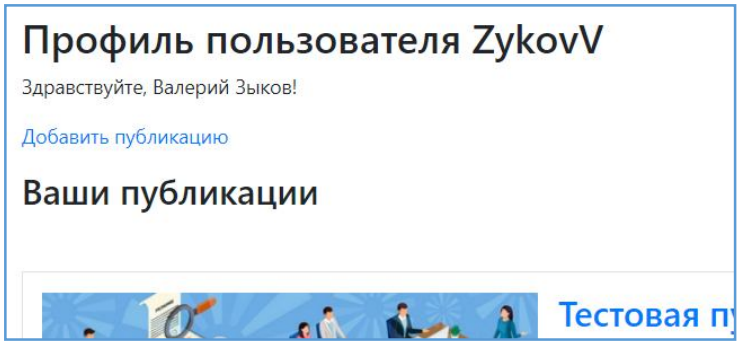



*Рисунок 31. Профиль. Добавление публикации.*

Внизу рядом с каждой публикацией имеется возможность либо - исправить либо - удалить публикацию.


*Рисунок 32. Страница пользовательских публикаций.*

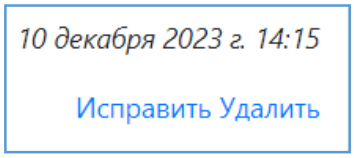



*Рисунок 33. Профиль. Исправление и удаление публикации.*

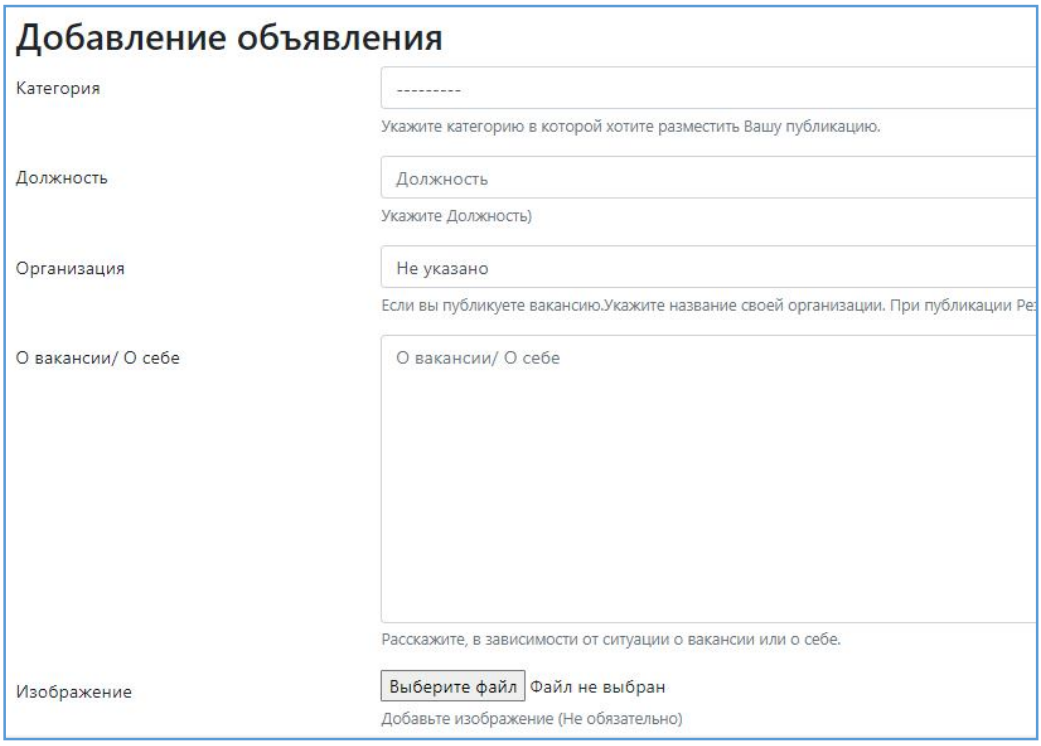



*Рисунок 34. Добавление публикации. Часть 1.*

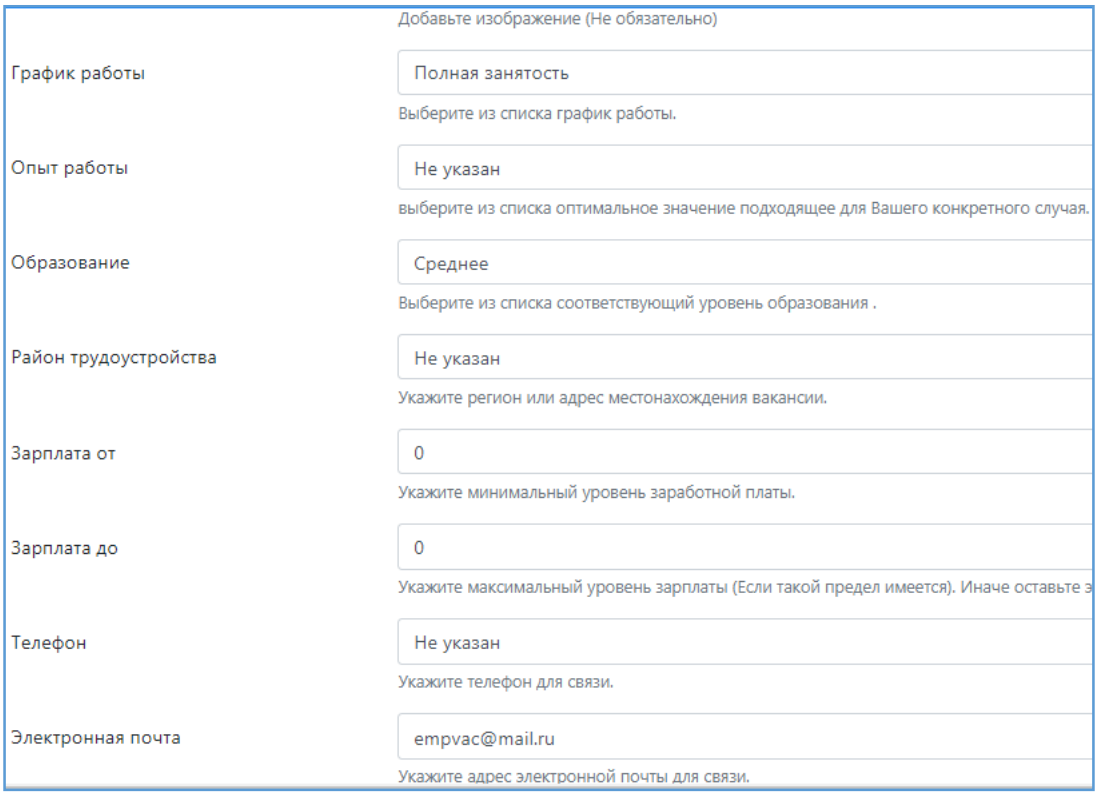



*Рисунок 35. Добавление публикации. Часть 2.*

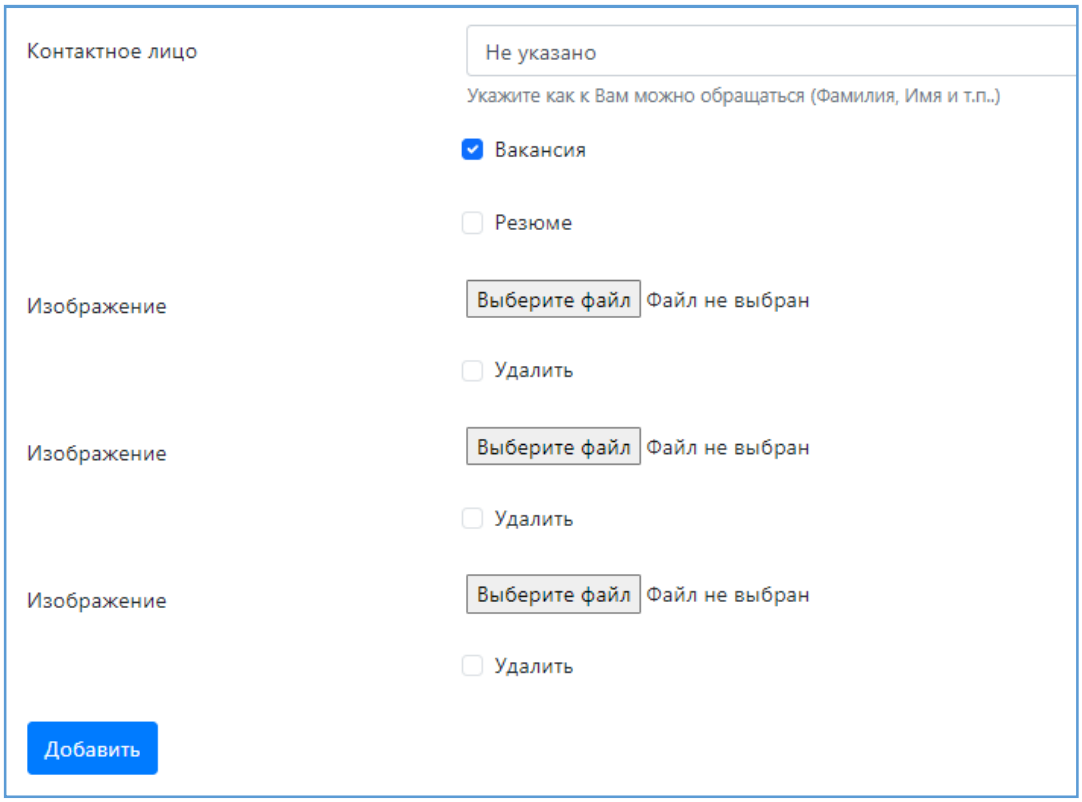



*Рисунок 36. Добавление публикации. Часть 3.*

*Таблица 2.*

*Схема получения данных от пользователя.*

| **Наименование поля** | **Способ передачи данных** | **Типы аккаунтов** | **Значение по умолчанию** |
|---|---|---|---|
| Категория | Выбор из списка | Все | |
| Должность | Заполняется вручную | Все | Не указано |
| Организация | Заполняется вручную | Только для вакансии | Не указано |
| О вакансии / О себе | Заполняется вручную | Все | Пустое поле |
| Изображение | Выбор файла на компьютере | Все | Если изображение не выбрано пользователем - подставляется изображение по умолчанию |
| График работы | Выбор из списка | Все | Полная занятость |
| Опыт работы | Заполняется вручную | Все | Не указан |
| Образование | Выбор из списка | Все | Среднее |
| Район трудоустройства | Заполняется вручную | Все | Не указан |
| Зарплата от | Заполняется вручную | Только для вакансии | 0 |
| Зарплата до | Заполняется вручную | Только для вакансии | 0 |
| Телефон | Заполняется вручную | Все | Не указан |
| Электронная почта | Автоматическая подстановка с возможностью изменения значения | Все | Электронная почта автора публикации указанная при регистрации |
| Контактное лицо | Ввести значение | Все | Не указано |
| Дополнительные изображения | Выбрать файл на компьютере (при необходимости) | Все | Если пользователь не выбрал ни одного дополнительного изображения, то ничего не выводится |

Для того чтобы не ограничивать пользователя в формате ввода данных – большую часть полей необходимо заполнить вручную. Поля, которые была возможность автоматизировать – автоматизированы. Такие поля как «График работы», «Образование», как правило, имеют стандартный набор значений поэтому предоставлена возможность выбора этих значений из списка. В поле «Электронная почта» значение подставляется из учётной записи пользователя, которая была указана при регистрации. Часть полей скрыта от пользователя, так как значения для этих полей формируются программно и участия пользователей не требуют. К примеру, пара полей: поле «Вакансия» и поле «Резюме», имеющих логический тип данных. Заполняются автоматически в зависимости от типа аккаунта пользователя, в дальнейшем эта информация используется для корректного отображения контента.

При необходимости значение любого поля доступного в форме можно изменить на любом этапе. Для этого пользователю необходимо в «Профиле» перейти к своим публикациям и под публикацией, которую необходимо изменить перейти по ссылке «Исправить».


*Рисунок 37. Изменение публикации.*

Процесс изменения публикации ничем не отличается от процесса добавления публикации. По завершению всех изменений необходимо нажать кнопку «Сохранить».

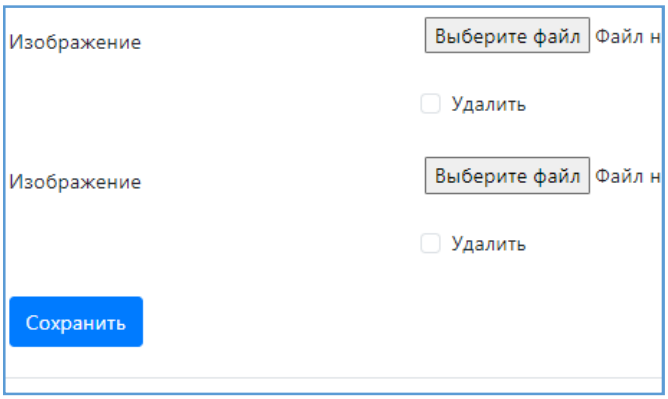



*Рисунок 38. Сохранение публикации.*

В случае если появится необходимость удалить публикацию – это можно сделать также здесь же в профиле в публикациях пользователя. В данном случае нужно выбрать под необходимой публикацией ссылку «Удалить».


*Рисунок 39. Удаление публикации.*

Форма добавления публикации для разных типов аккаунта не отличается, но при формировании вывода публикации на страницу сайта вид отображения вакансии и резюме отличается.

В случае вывода «Резюме» отображаются не все поля, которые представлены в «Вакансии». К примеру, поле «Организация» не имеет никакого смысла

в «Резюме», так как в резюме пользователь рассказывает о себе, являясь физическим лицом и никакой привязки к какой-либо организации в данном случае нет. Также наименования некоторых полей отличаются, так как они имеют разный контекст.

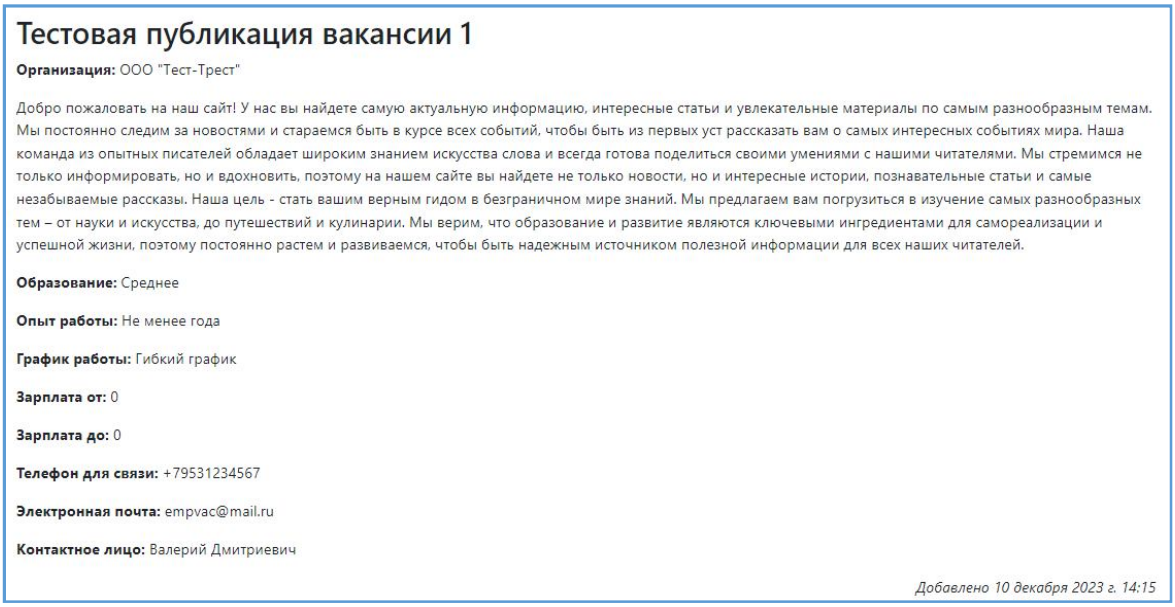



*Рисунок 40. Публикация «Вакансия».*

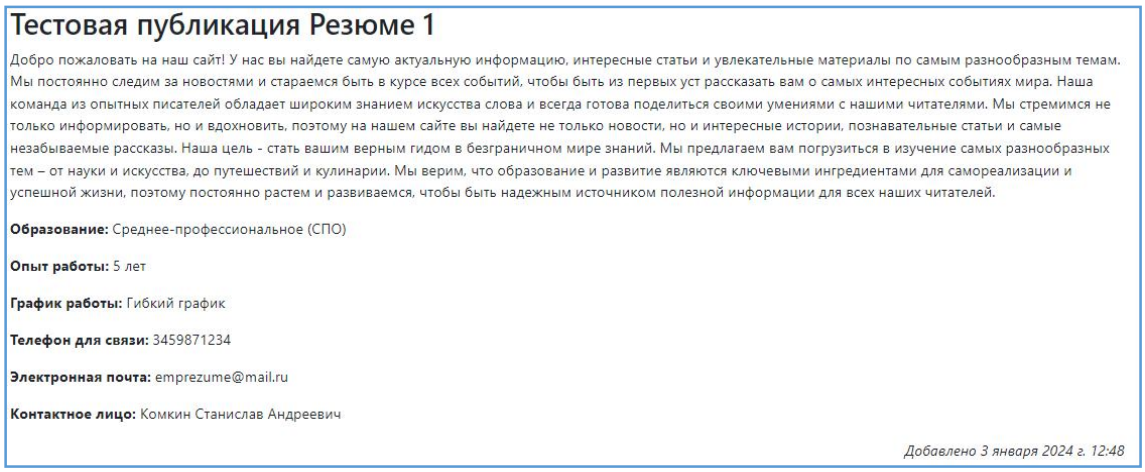



*Рисунок 41. Публикация «Резюме».*

Возле каждой публикации на сайте есть возможность оставить комментарий. При этом у зарегистрированного пользователя поле «Автор» заполнится автоматически. В это поле подставляется имя текущего пользователя.

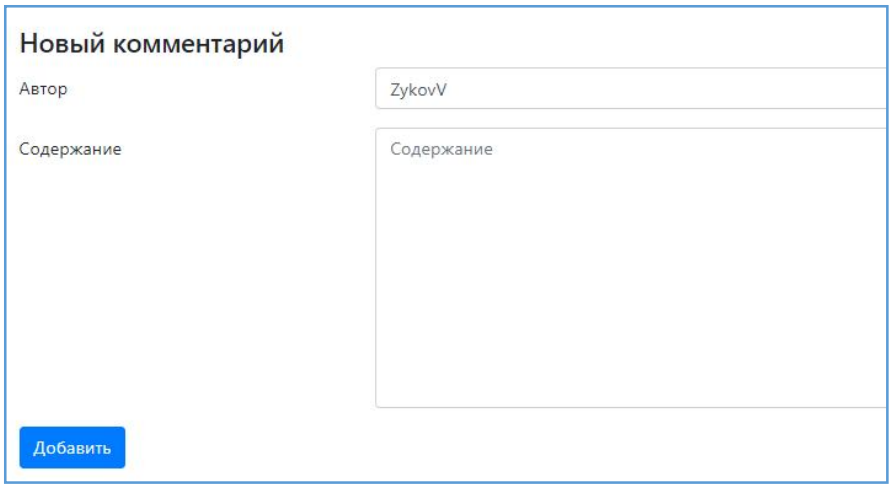



*Рисунок 42. Добавление нового комментария.*

При необходимости пользователь может изменить в своём профиле: личные данные или пароль. Также присутствует возможность удаления своего аккаунта.

Для изменения личных данных пользователю нужно перейти по ссылке «Изменить личные данные».

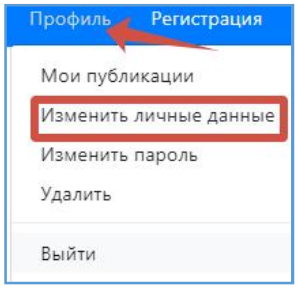



*Рисунок 43. Ссылка, ведущая на страницу изменения личных данных пользователя.*

На странице изменения личных данных пользователя, реализована возможность изменения любых данных пользователя заданных при регистрации, за исключением пароля.

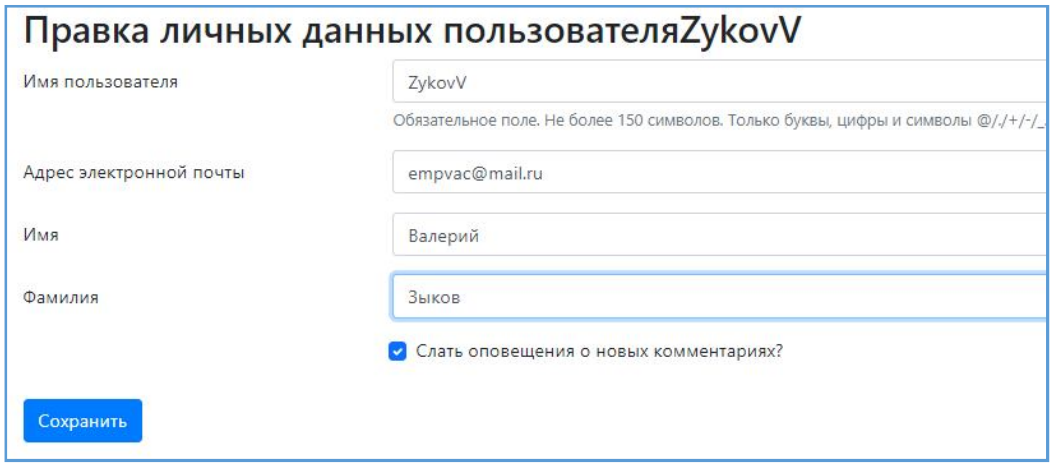



*Рисунок 44. Страница правки личных данных пользователя.*

Для изменения пароля необходимо перейти по ссылке «Изменить пароль» в профиле пользователя.

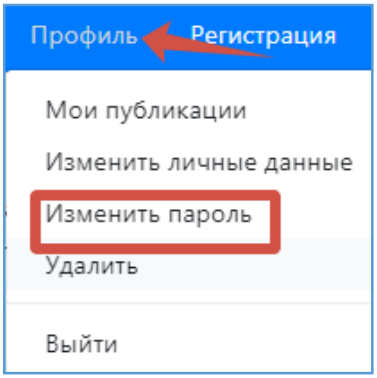



*Рисунок 45. Ссылка на страницу изменения пароля пользователя.*

На странице изменения пароля пользователя необходимо ввести старый, то есть текущий, пароль и дважды ввести новый пароль.

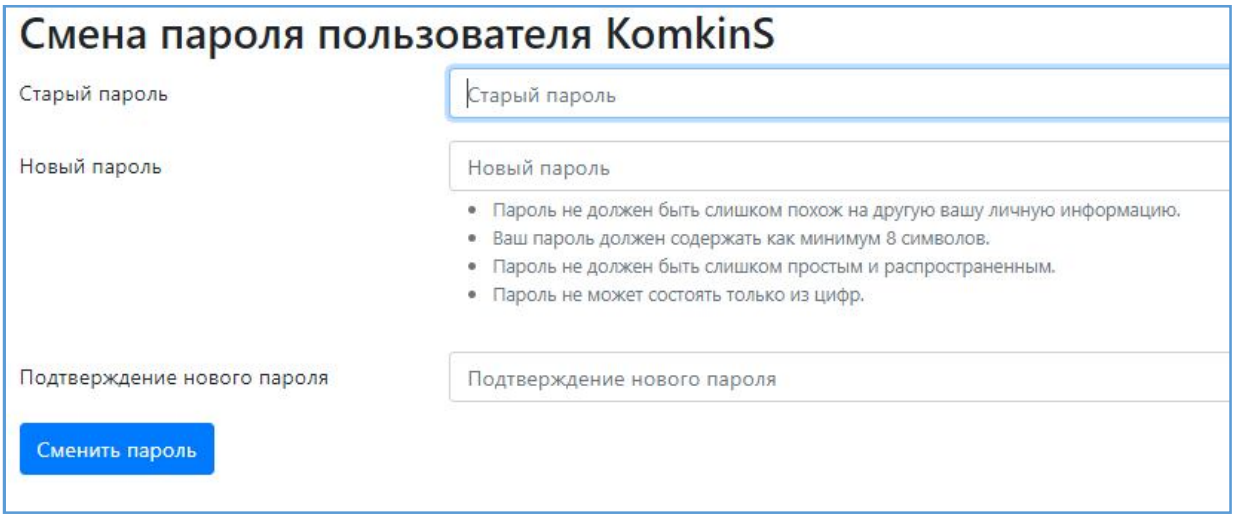



*Рисунок 46. Страница смены пароля пользователя.*

Для изменения личных данных пользователю нужно перейти по ссылке «Изменить личные данные».

При необходимости удалить аккаунт - это можно также сделать через профиль пользователя, посредством перехода по ссылке «Удалить».

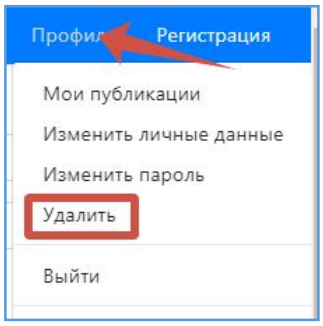



*Рисунок 47. Ссылка, ведущая на страницу удаления аккаунта пользователя.*

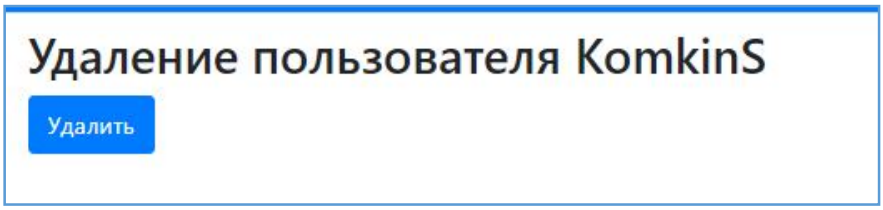



*Рисунок 48. Страница удаления аккаунта пользователя.*

На сайте также реализована административная панель, вход в которую доступен ограниченному числу пользователей, назначаемых администратором сайта.

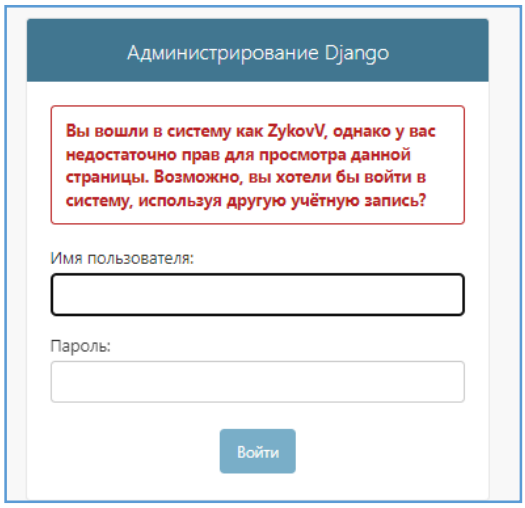



*Рисунок 49. Вход в АдминПанель пользователя, не имеющего на это прав.*

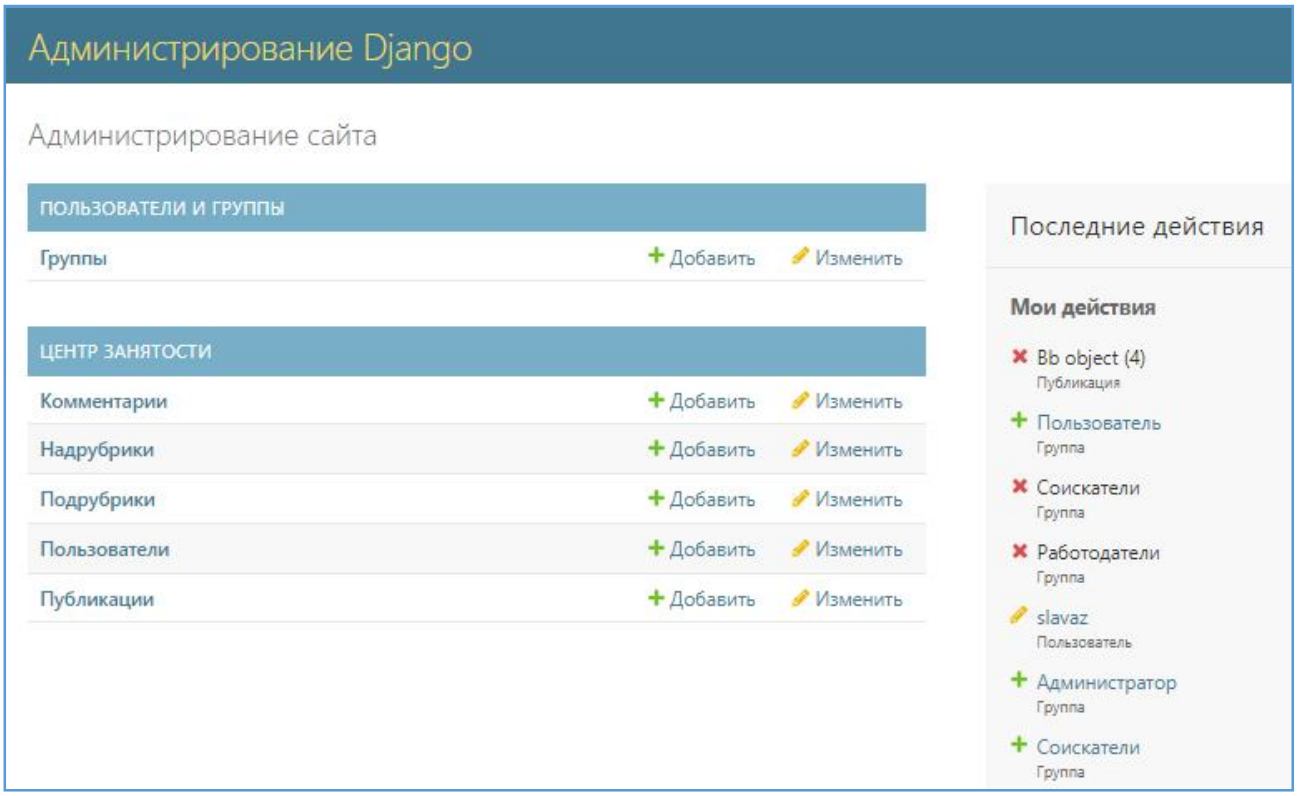



*Рисунок 50. Главная страница Административной панели.*

На главной странице Административной Панели отображаются все доступные веб-приложения. Приложение «Пользователи и группы» это встроенное по умолчанию приложение Django, оно присутствует всегда, даже если проект только что создан. В этом приложении реализована возможность управлять правами пользователей,объединяя их в определённые группы, чтобы впоследствии, применить к ним определённый набор прав.

Приложение «Центр занятости» позволяет управлять всем контентом, размещённым на сайте: комментариями, рубриками (категориями), публикациями.

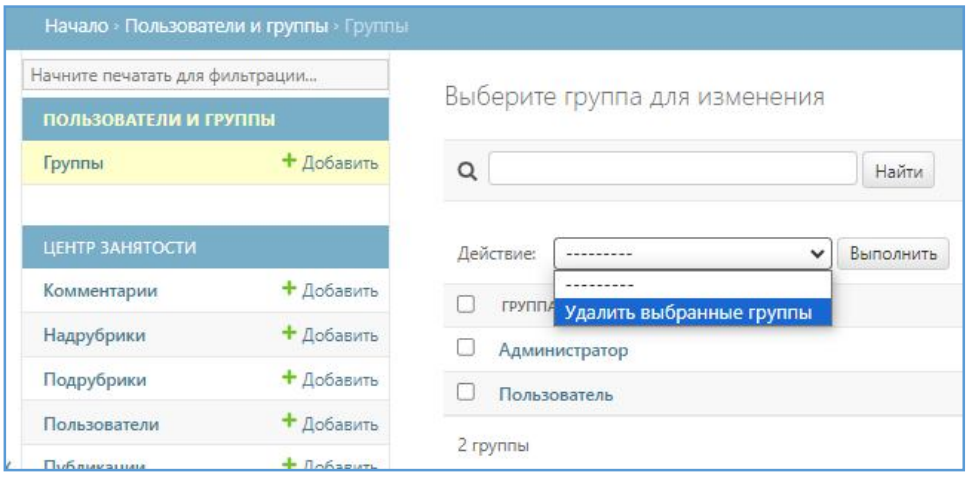



*Рисунок 51. Группы (приложение «Пользователи и Группы»)*

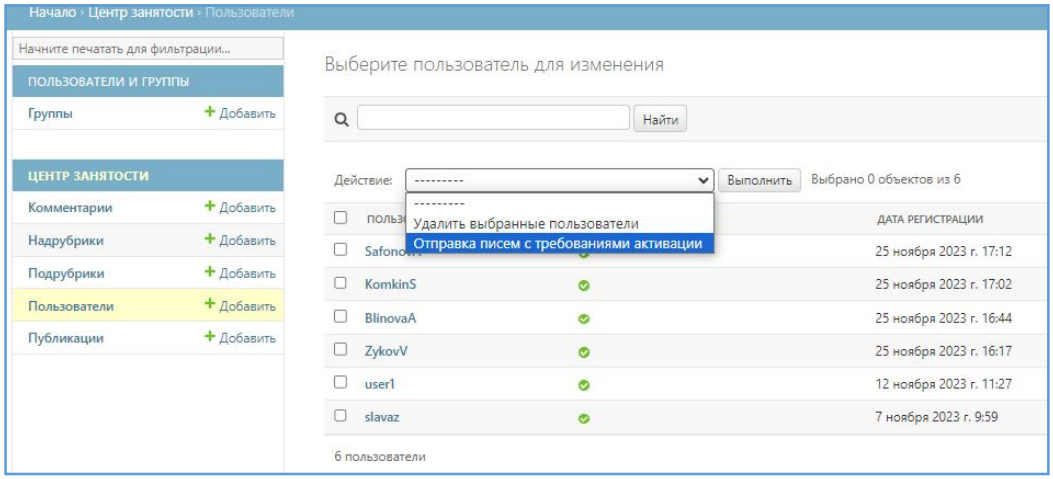



*Рисунок 52. Пользователи (приложение «Центр занятости»).*

В Административной панели в разделе «Пользователи» реализована возможность управления пользователями. Здесь можно удалить выбранных пользователей или отправить им письмо с требованием активации. В момент регистрации нового пользователя на сайте такое письмо автоматически отправляется на электронную почту указанную пользователем. В случае если пользователь в течении какого-то промежутка времени не активировал свой аккаунт, можно отправить письмо с требованием активации повторно. Для обнаружения таких

пользователей в общей массе всех пользователей сайта - предусмотрен специальный фильтр, посредством которого можно вывести пользователей которые:

- Прошли активацию;
- Не прошли более 3 дней;
- Не прошли более недели;
- Все пользователи;

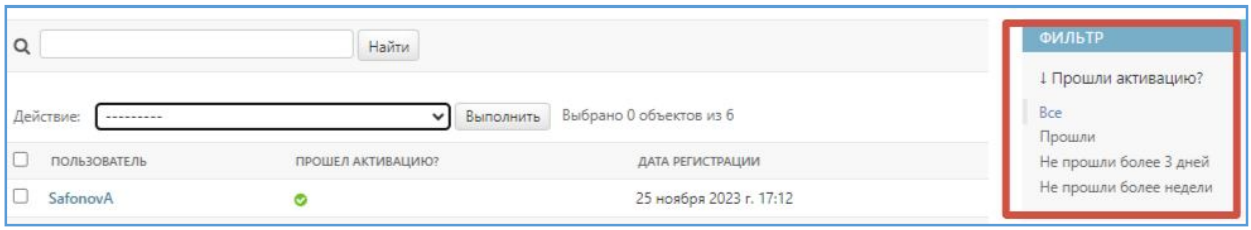



*Рисунок 53. Фильтрация пользователей.*

В разделе «Публикации» администратор может просматривать публикации любого из пользователей, и при необходимости удалять их.

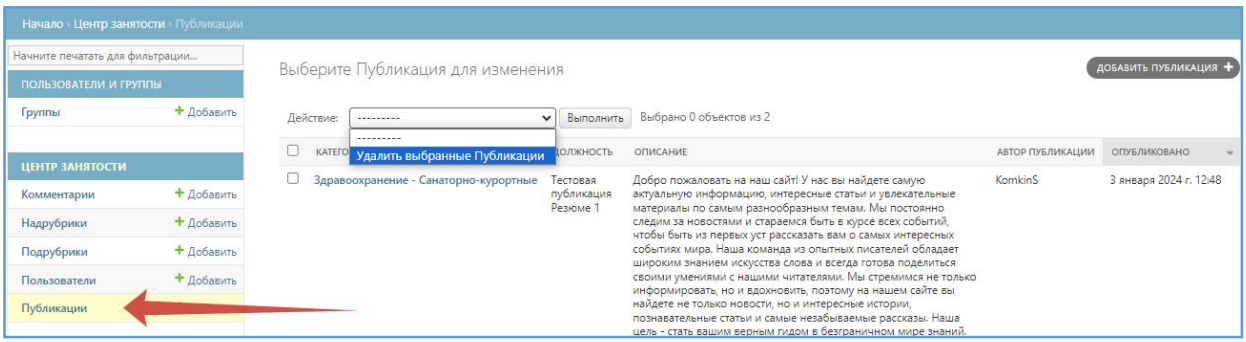



*Рисунок 54. Публикации пользователей.*

Работа с другими разделами административной панели реализована подобным образом.

Программная часть данной информационной системы реализована посредством фреймворка Django. В данном фреймворке используется архитектурный паттерн Model-View-Template (MVT). Схематично эту архитектуру можно представить следующим образом:

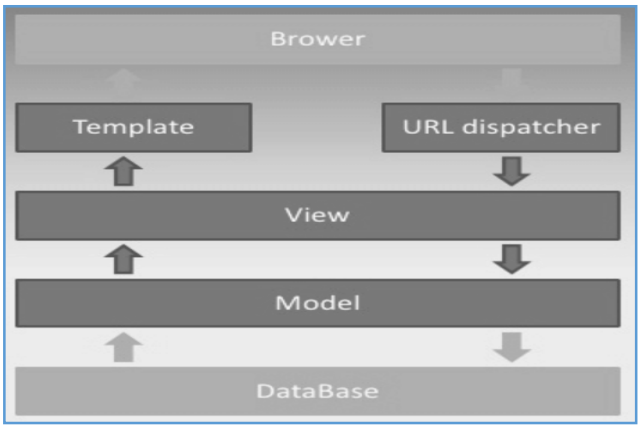



*Рисунок 55. Архитектурный паттернModel-View-Template (MVT).*

Данный паттерн имеет следующие элементы:

**URLdispatcher**: в момент получения запроса, на основании адреса URL который был запрошен, данный диспетчер определяет, какой ресурс должен обрабатывать этот запрос. При поступлении любого нового запроса от клиента Django извлекает из целевого интернет-адреса путь, который в последствии передаёт маршрутизатору. Маршрутизатор последовательно сравнивает этот путь с шаблонными путями из существующего списка маршрутов. Как только будет найдено совпадение, управление передаётся контроллеру, который связан с совпадающим шаблонным путём.

```
path('', index, name='index'),   # Главная страница
```

*Рисунок 56. Маршрут для вывода главной страницы.*

Данный маршрут передаёт управление контроллеру «index», который выводит на экран главную страницу.

```
path('<str:page>/', other_page, name='other'),  # Вспомогательные страницы
```

*Рисунок 57. Маршрут для вывода вспомогательных страниц.*

Показанный маршрут передаёт управление контроллеру, который позволяет выводить на экран все вспомогательные страницы.

Все маршруты которые были применены в проекте, находятся в приложении См.Приложение 2. Модуль Urls.py.

**View**: Контроллер (представление) получает запрос, обрабатывает его соответствующим образом и отправляет ответ пользователю. В том случае если для выполнения запроса есть необходимость обращения к какой-либо модели или базе данных, то контроллер взаимодействует с ними. Для формирования ответа можно применять шаблоны (Template). Контроллер может представлять собой как функцию, так и класс. Функции более универсальны, но иногда трудоёмки в программировании. Классы позволяют выполнять типовые задачи, наподобие вывода списка каких-либо позиций, минимумом кода. Контроллеры объявляются в модуле views.рупакета приложения.

```python
# Контроллер для главной страницы
def index(request):
    bbs = Bb.objects.filter(is_active=True)[:10]
    context = {'bbs': bbs}
    return render(request, 'main/index.html', context)
```

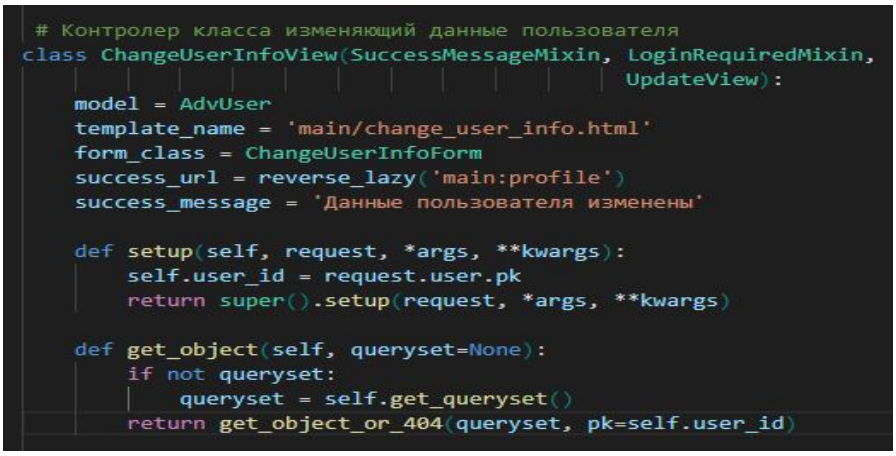
*Рисунок 58. Контроллер, выводящий Главную страницу.*

Данный контроллер выводит Главную страницу, которая отображает 10 последних публикаций на сайте.

```python
# Контроллер для вспомогательных страниц
def other_page(request, page):
    try:
        template = get_template('main/' + page + '.html')
    except TemplateDoesNotExist:
        raise Http404
    return HttpResponse(template.render(request=request))
```

*Рисунок 59. Контроллер, выводящий вспомогательные страницы.*

Контроллер, который выводит все вспомогательные страницы. Имя выводимой страницы получаем из входного параметра **page**, добавляем к нему разрешение и путь, получая тем самым полный путь к целевому шаблону. Пробуем загрузить этот шаблон вызовом функции **get_template().** Если процесс загрузки прошёл успешно, то на основе этого шаблона формируем результирующую страницу. В случае если шаблон не загрузился, возбуждается исключение **TemplateDoesNotExist**, которое мы перехватываем и возбуждаем другое исключение **Http404**, которое отправляет страницу с сообщением об ошибке 404 (запрошенная страница не существует).



```
# Контролер класса изменяющий данные пользователя
class ChangeUserInfoView(SuccessMessageMixin, LoginRequiredMixin,
                         UpdateView):
    model = AdvUser
    template_name = 'main/change_user_info.html'
    form_class = ChangeUserInfoForm
    success_url = reverse_lazy('main:profile')
    success_message = 'Данные пользователя изменены'

    def setup(self, request, *args, **kwargs):
        self.user_id = request.user.pk
        return super().setup(request, *args, **kwargs)

    def get_object(self, queryset=None):
        if not queryset:
            queryset = self.get_queryset()
        return get_object_or_404(queryset, pk=self.user_id)
```

*Рисунок 60. Контроллер класса, изменяющий данные.*

В процессе работы данный контроллер извлекает из модели **AdvUser** запись, которая представляет текущего пользователя. Для этого предварительно необходимо получить из объекта текущего пользователя его ключ, который хранится в атрибуте **user**.

Все контроллеры, которые были определены в рамках проекта находятся в приложении См. Приложение 3. Модуль Views.py.

**Model**: Модель описывает данные, которые используются в приложении. Каждый отдельный класс, как правило, соответствует отдельной таблице в базе данных. Модель по своей сути это класс, который описывает определённую таблицу в базе данных, точнее набор имеющихся в таблице полей. Каждый отдельный экземпляр класса модели представляет собой отдельную запись таблицы. Модели объявляются в модуле models.py пакета приложения. Стандартная модель пользователя встроенная в фреймворк нам не подходит, так как нам необходимо хранить дополнительные данные о пользователе. Создадим на основе абстрактного класса **AbstractUser** свой собственный класс **AdvUser.**

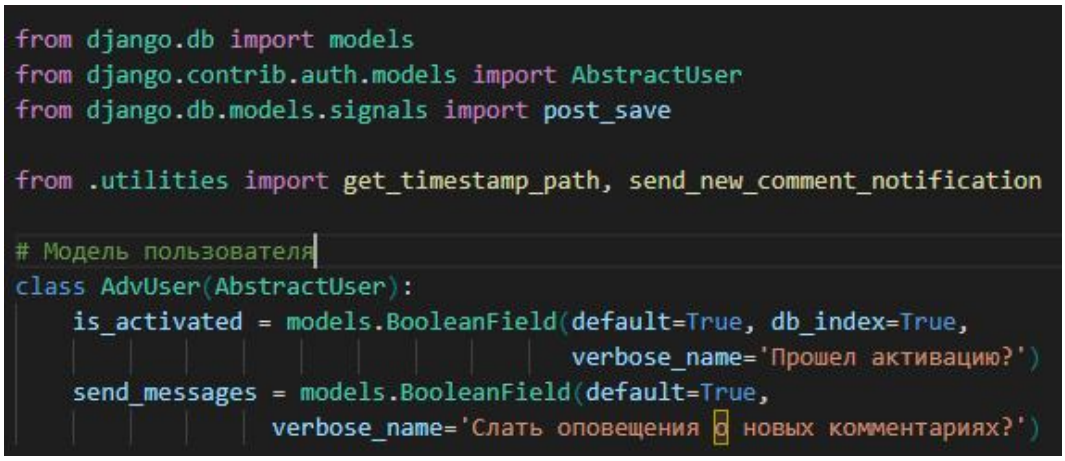



*Рисунок 61. Модель пользователя AdvUser*

Все основные поля для этого класса наследуем у класса **AbstractUser**, расширим этот класс, добавив в него два новых поля: is_activated, которое будет указывать на то что пользователь прошёл активацию, и поле send_messages– если значение этого поля будет истинным, то на электронную почту автора публикации будет приходить оповещение о том, что под его публикацией появился новый комментарий.

Все модели, которые были определены в рамках проекта находятся в приложении См. Приложение 4. Модуль Models.py.

**Template**: Представляет собой логику контроллера (представления) в виде сгенерированной разметки html. В шаблоне формируется внешний вид приложения. Шаблон представляет специальный синтаксис, позволяющий вставить данные в HTML-код.

В рамках разрабатываемой информационной системы формируем базовый шаблон base.html. Этот шаблон описывает все элементы, которые мы хотим видеть на страницах нашего сайта.

В файле **basic.html**(См. Приложения) посредством html– разметки описана вся структура страницы: меню, элементы поиска и т.д. Все остальные шаблоны в своём теле наследуют содержимое этого файла и содержат только сам контент. Всё оформление и элементы подгружаются из родительского файла basic.html.

Исходный код некоторых html-шаблонов которые были применены в рамках проекта находится в приложении См.Приложение 1. Шаблоны(Templates).

```
{% extends "layout/basic.html" %}

{% load bootstrap4 %}

{% block title %}Вход{% endblock %}

{% block content %}
<h2>Вход</h2>
{% if user.is_authericated %}
<p>Вы уже выполнили вход.</p>
{% else %}
<form method="post">
    {% csrf_token %}
    {% bootstrap_form form layout='horizontal' %}
    <input type="hidden" name="next" value="{{next}}">
    {% buttons submit='Войти' %}{% endbuttons %}
</form>
{% endif %}
{% endblock %}
```

*Рисунок 62. Шаблон страницы входа на сайт.*

```
{% extends "layout/basic.html" %}

{% load bootstrap4 %}

{% block title %}Регистрация{% endblock %}

{% block content %}
<h2>Регистрация нового пользователя</h2>
<form method="post">
    {% csrf_token %}
    {% bootstrap_form form layout='horizontal' %}
    {% buttons submit='Зарегистрироваться' %}{% endbuttons %}
</form>
{% endblock %}
```

*Рисунок 63. Шаблон страницы регистрации пользователя*

**Forms:** Непосредственно для реализации ввода данных в Django используются формы. Формы – это объекты, которые предназначены для вывода на страницу веб-формы с необходимыми управляющими элементами, а также для проверки введённых данных на корректность. Если форма связана с моделью, она может сохранить самостоятельно данные в базе данных. Форма обычно объявляется в файле **forms.py**уровня приложения в котором она будет исполь-

зоваться. Объявление формы **ChangeUserInfoForm**, связанной с моделью **AdvUser**, предназначенной для ввода основных данных выглядит следующим образом.

```
class ChangeUserInfoForm(forms.ModelForm):
    email = forms.EmailField(required=True, label='Адрес электронной почты')

    class Meta:
        model = AdvUser
        fields = ('username', 'email', 'first_name', 'last_name',
                  'send_messages')
```

*Рисунок 64. Форма для ввода основных данных.*

Делаем полное объявление поля **email**, так как мы хотим сделать его обязательным для заполнения. А так как атрибуты других полей у нас не меняются, то для этих полей делаем быстрое объявление.

Все формы, которые были определены в рамках проекта находятся в приложении См. Приложение 5. Модуль Forms.py.

Для реализации отправки писем пользователям с требованием активации создаём в пакете приложения новый модуль **utilities.py** и объявляем в нём функцию **send_activation_notification**.

```
signer = Signer()

# Функция для рассылки электронных писем
def send_activation_notification(user):
    if ALLOWED_HOSTS:
        host = 'http://' + ALLOWED_HOSTS[0]
    else:
        host = 'http://localhost:8000'
    context = {'user': user, 'host': host, 'sign': signer.sign(user.username)}
    subject = render_to_string('email/activation_letter_subject.txt', context)
    body_text = render_to_string('email/activation_letter_body.txt', context)
    user.email_user(subject, body_text)
```

*Рисунок 65. Функция send_activation_notification.*

Для формирования интернет-адреса, который будет вести на страницу подтверждения активации. Необходимо получить определённые данные:

- Домен, на котором находится наш сайт;

- Некоторое значение, которое может уникально идентифицировать только что зарегистрированного пользователя и при этом устойчивое к попыткам его изменить;

Домен мы можем получить из файла конфигурации проекта (**settings.py**), из параметра **ALLOWED_HOST**(список разрешённых доменов). В нашем случае выбираем самый первый домен, который присутствует в списке. В том случае, если список доменов пуст, задействуем интернет-адрес, который используется отладочным сервером Django.

В качестве стойкого к подделке уникального идентификатора будет применяться его имя, защищённое цифровой подписью. Создание цифровой подписи осуществляется посредством класса **Signer**.

Для формирования тела письма и текста темы будут использованы текстовые шаблоны.

**Инструменты для администрирования пользователей.**

В редактор, посредством которого администрация сайта сможет работать с пользователями, имеющими регистрацию добавим определённый функционал:

- Возможность фильтрации пользователей по именам, адресам электронной почты, уже выполнивших активацию, не выполнивших её в течение трёх дней и недели;
- Действие по отправке выбранным пользователям писем с требованием пройти активацию.

```
def send_activation_notifications(modeladmin, request, queryset):
    for rec in queryset:
        if not rec.is_activated:
            send_activation_notification(rec)
    modeladmin.message_user(request, 'Письма с требованиями отправлены')
send_activation_notifications.short_description = \
'Отправка писем с требованиями активации'

class NonactivatedFilter(admin.SimpleListFilter):
    title = 'Прошли активацию?'
    parameter_name = 'actstate'

    def lookups(self, request, model_admin):
        return (
                    ('activated', 'Прошли'),
                    ('threedays', 'Не прошли более 3 дней'),
                    ('week', 'Не прошли более недели'),
                )
def queryset(self, request, queryset):
    val = self.value()
    if val == 'activated':
        return queryset.filter(is_active=True, is_activated=True)
    elif val == 'threedays':
        d = datetime.date.today() - datetime.timedelta(days=3)
        return queryset.filter(is_active=False, is_activated=False,
                               date_joined__date__lt=d)
    elif val == 'week':
        d = datetime.date.today() - datetime.timedelta(weeks=1)
        return queryset.filter(is_active=False, is_activated=False,
                               date_joined__date__lt=d)

class AdvUserAdmin(admin.ModelAdmin):
    list_display = ('__str__', 'is_activated', 'date_joined')
    search_fields = ('username', 'email', 'first_name', 'last_name')
    list_filter = (NonactivatedFilter,)
    fields = (('username', 'email'), ('first_name', 'last_name'),
              ('send_messages', 'is_active', 'is_activated'),
              ('is_staff', 'is_superuser'),
              'groups', 'user_permissions',
              ('last_login', 'date_joined'))
    readonly_fields = ('last_login', 'date_joined')
    actions = (send_activation_notifications,)

admin.site.register(AdvUser, AdvUserAdmin)
```

*Рисунок 66. Код класса редактора AdvUserAdmin*

Код класса редактора AdvUserAdmin будет находится в модуле admin.py В списке записей выводим строковое представление записи (имя пользователя – как реализовано в модели **AbstractUser**, от которого была наследована модель **AdvUser**), поле признака, выполнил ли пользователь активацию, фиксированная дата его регистрации. Дополнительно даём разрешение производить фильтрацию по полям имени, адреса электронной почты, имени и фамилии.

Для фильтрации пользователей, выполнивших активацию, не выполнивших её в течение трёх дней и недели, используем класс **NonactivatedFilter**.

Указываем явно список полей, которые должны выводиться в формах для правки пользователей, чтобы построить их в удобном для работы порядке. Поля даты последнего входа на сайт и регистрации сделаем доступными только для чтения.

Регистрируем действие, которое будет рассылать пользователям письма с предписаниями выполнить активацию. Это действие реализовано посредством функции **send_activation_notification()**, которая была ранее объявлена в модуле **utilities.py** и непосредственно производящую отправку писем.

Все инструменты для администрирования сайта которые были определены в рамках проекта находятся в приложении См.Приложение 6. Модуль Admin.py.

## 2.3 Результаты апробации

По завершению разработки информационной системы для центра занятости система была развёрнута на локальном компьютере (ноутбуке), и была показана группе студентов института математики, физики и информатики, у которых была возможность протестировать информационную систему. Итоги апробации определялись путём экспертного оценивания: участникам тестирования (апробации) была выдана анкета с рядом критериев, каждый из которых можно оценить по 5-бальной шкале, где 0 – «плохо», а 5 – «отлично».

Критерии, представленные в анкете, были следующие:

1.Удобство работы

2.Корректность работы

3.Реализация модуля регистрации

4.Реализация модуля комментариев

5.Реализация пользовательского профиля

6.Реализация добавления публикации

7.Реализация административной панели

8.Удовлетворение реализованной информационной системой техническому заданию.

Результаты апробации представлены в Таблице 3. Согласованность мнений экспертов по каждому из критериев выражается коэффициентом вариации $C_v$ .

Таблица 3

Экспертные оценки, уровень согласованности экспертов

| Критерий | Номер эксперта и его оценка | | | | | | Среднее значение | Степень согласованности |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | | |
| Удобство работы | 5 | 5 | 5 | 5 | 5 | 5 | 5 | Высокая |
| Корректность работы | 5 | 5 | 5 | 5 | 5 | 5 | 5 | Высокая |
| Реализация модуля регистрации | 4 | 5 | 5 | 5 | 5 | 5 | 4,8 | Высокая |
| Реализация модуля комментариев | 5 | 5 | 4 | 5 | 5 | 5 | 4,8 | Высокая |
| Реализация пользовательского профиля | 5 | 5 | 5 | 4 | 4 | 5 | 4,7 | Высокая |
| Реализация добавления публикации | 5 | 5 | 5 | 5 | 5 | 5 | 5 | Высокая |
| Реализация административной панели | 4 | 5 | 5 | 4 | 5 | 5 | 4,7 | Высокая |
| Удовлетворение | 5 | 5 | 5 | 5 | 5 | 5 | 5 | Высокая |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| реализованной информационной системой техническому заданию | | | | | | | | |

Среди замечаний и рекомендаций экспертов по улучшению информационной системы для центра занятости было следующие: «Не доработанное боковое меню со списком сфер деятельности».

Таким образом, по результатам апробации, разработку системы можно считать завершённой и готовой к использованию по прямому назначению.

# Заключение

При выполнении данной выпускной работы была спроектирована и реализована информационная система для службы занятости. Данная информационная система является веб-приложением, реализованном на фреймворке Django. При написании приложения был использован язык программирования Python. Согласно сформулированным заданиям в процессе выполнения работы было проделано следующее:

1. Проведён анализ текущей ситуации на рынке рекрутинговых организаций, а также существующих информационных систем занимающихся данной тематикой. В результате данной аналитики была обоснована актуальность информационной системы для службы занятости.
2. Проведена глубокая аналитика технологий и методов применяемых при разработке современных информационных систем. В результате были определены оптимальные технологии для реализации информационной системы для службы занятости.
3. На основе проведённой ранее аналитики было подготовлено техническое задание на разработку информационной системы для службы занятости.
4. Опираясь на разработанное техническое задание была спроектирована, а в последствии и реализована информационная система для службы занятости.

Таким образом, следует считать, что результаты разработки соответствуют всем требованиям технического задания, работа носит законченный характер, цель работы достигнута и задачи выполнены.

# Список информационных источников


1. Обзор сайта rabota.ru // naim.ru Поиск работы и сотрудников в ритейле : сайт. – URL: https://www.naim.ru/reviews/обзор-сайта-rabota-ru-00227.html?ysclid=lrj0a2q1fv266987961 (дата обращения: 18.01.2024)

2. Выбор case-средств // StudFiles : сайт. – URL: https://studfile.net/preview/6469081/page:8/ (дата обращения: 19.01.2024)

3. ГОСТ Р ИСО/МЭК 9126-93 : дата введения 06.30.1994. – Москва : ГОССТАНДАРТ РОССИИ, 1994. – 12 с.

4. Что такое Django // METANIT.COM : сайт. – URL: https://metanit.com/python/django/1.1.php (дата обращения: 23.01.2024)

5. Введение в стили // METANIT.COM : сайт. – URL: https://metanit.com/web/html5/5.1.php (дата обращения: 24.01.2024)

6. Подключение фреймворка Bootstrap к сайту // ИТШЕФ : сайт. – URL: https://itchief.ru/bootstrap/installation (дата обращения: 24.01.2024)

7. Маклаков С.В. Создание информационных систем с All Fusion Modeling Suite. – М.: ДИАЛОГ – МИФИ, 2013. – 224с.

8. Маклаков С.В. BPWin и ERWin. CASE – средства разработки информационных систем. – М.: ДИАЛОГ – МИФИ, 2013. – 256с.

9. Цикритизис Д., Лоховски Ф. Модели данных. – М.: Финансы и статистика, 2014. – 344 с.

10. Атре Ш. Структурный подход к организации баз данных. – М.:Финансы и статистика, 2014. – 320 с.

11. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. – М.: Финансы и статистика, 2014. – 351 с.

12. Мартин Дж. Планирование развития автоматизированных систем. – М.: Финансы и статистика, 2014. – 196 с.

13. Мейер М. Теория реляционных баз данных. – М.: Мир, 2013. – 608 с.

14. Грекул, В.И. Проектирование информационных систем. М.: Бином. Лаборатория знаний, 2007.

15. Вигерс К. Битти Д. Разработка требований к программному обеспечению. М.: Русская редакция, 2014.

16. Зараменских Е.П. Управление жизненным циклом информационных систем: монография. Новосибирск: Издательство ЦРНС, 2014.

17. Кожевников П.А. Моделирование и оптимизация бизнес-процессов Call-Центра. Курсовая работа М., 2016.

18. Когаловский М. Р. Перспективные технологии информационных систем. М.: ДМК Пресс; М: Компания АйТи, 2003.

19. Ю. А. Маглинец. Анализ требований к автоматизированным информационным системам. М.: Бином, 2008.

20. Калянов Г.Н. Моделирование, анализ, реорганизация и автоматизация бизнес-процессов: Учебное пособие. М.: Финансы и статистика, 2006.

21. Автоматизированные Системы Стадии создания. ГОСТ 34.601-90 Комплекс стандартов на автоматизированные системы. М.: ИПК издательство стандартов, 1997.

22. Коберн А. Современные методы описания функциональных требований к системам. М.: Лори, 2011

23. Васильев, Р. Б. Стратегическое управление информационными системами. М.: Интернет-Университет Информационных Технологий, 2010.

24. Фуфаев Д.Э. Разработка и эксплуатация автоматизированных информационных систем: учебник для студ. учреждений сред. проф. Образования. М.: Издательский центр "Академия", 2014.

25. Гвоздева Т.В. Проектирование информационных систем: учеб. Ростов н/Д.: Феникс, 2009.

26. Елиферов В.Г. Бизнес-процессы: регламентация и управление. М.: ИНФРА-М, 2008.

27. Дронов В.А. Django 3.0. Практика создания веб-сайтов на Python. - СПб.: БХВ-Петербург, 2023. - 704 с.

28. Кольцов Д.М., Дубовик Е.В. Справочник PYTHON. Кратко, быстро, под рукой. - СПб.: Издательство Наука и Техника, 2022. - 288 с.

29. PostgreSQL. Основы языка SQL: учебно-практическое пособие / Моргунов Е.П., Под ред. Рогова Е.П., Лузанова П.В. - СПб.: БХВ-Петербург, 2021. - 336 с.

30. Руководство по языку программирования Python // METANIT.COM URL: https://metanit.com/python/tutorial/ (дата обращения: 07.01.2024).

31. Руководство по веб-фреймворку Django // METANIT.COM URL: https://metanit.com/python/django/ (дата обращения: 07.01.2024).

32. Руководство по PostgreSQL // METANIT.COM URL: https://metanit.com/sql/postgresql/ (дата обращения: 07.01.2024).

33. Руководство по HTML5 и CSS3 // METANIT.COM URL: https://metanit.com/web/html5/ (дата обращения: 07.01.2024).

34. Документация по фреймворку и библиотекам Django // DJANGO.FUN URL: https://django.fun/docs/ (дата обращения: 07.01.2024).

35. Документация Bootstrap на русском языке // Bootstrap.ru URL: https://bootstrap-4.ru/docs/5.3/getting-started/introduction/ (дата обращения: 07.01.2024).

# Приложения

## Приложение 1. Исходный код Html-шаблоновTemplates (выборочно)

Исходный код базового шаблона **basic.html**

```
{% load bootstrap4 %}
{% load static %}
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha2/dist/css/bootstrap.min.css"
      rel="stylesheet" integrity="sha384-
aFq/bzH65dt+w6FI2ooMVUpc+21e0SRygnTpmBvdBgSdnuTN7QbdgL+OapgHtvPp"
        crossorigin="anonymous">
<title>{% block title %}Главная{% endblock %} – Центр занято-
сти</title>
        {% bootstrap_css %}   <!—Привязка к странице таблицы-
стилей Bootstrap -->
<link rel="stylesheet" type="text/css" href="{% static
'main/style.css' %}">
<link href="album.css" rel="stylesheet">
        {% bootstrap_javascript jquery='slim' %}
</head>
<body>
<nav class="navbar navbar-expand-md sticky-top navbar-dark
bg-primary pb-1 pt-1"><!—Основные настройки отображения панели  --
>
```

```
     <a class="navbar-brand pl-5" href="{% url 'main:index'
%}">Центрзанятости</a>
     <button class="navbar-toggler">
     <span class="navbar-toggler-icon" data-toggle="collapse" da-
ta-target="#navbar"></span>
     </button>
     <div id="navbar" class="collapse navbar-collapse justify-
content-between">
     <form class="form-inline">
     <div class="input-group">
     <input type="text" placeholder="Search">
     <div>
     <button class="btn-dark" type="submit">Отправить</button>
     </div>
     </div>
     </form>
     <ul class="navbar-nav">
     <li class="nav-item dropdown">
     <a class="nav-link text-light   px-3 "
                  data-toggle="dropdown" href="{% url
'main:show_vacancy' %}">Вакансии</a>

     <div><!--class="dropdown-menu">-->
     </div>
     </li>
     <li class="nav-item dropdown">
     <a class="nav-link text-light  px-3"
              href="{% url 'main:show_resume' %}">Резюме</a>
     <div class="dropdown-menu">
                  {% for super_rubric in super_rubrics %}
     <a class="dropdown-item" href="{% url 'main:by_rubric'
pk=super_rubric.pk %}">{{ super_rubric.name }}</a>
     {% endfor %}
     </div>
```

```
    </li>
    {% if user.is_authenticated %}
    <li class="nav-item dropdown">
    <a class="nav-link text-light  px-3" data-toggle="dropdown"
                href="#" role="button" aria-haspopup="true" aria-
expanded="false">Профиль</a>
    <div class="dropdown-menu">
    {% if user.is_staff %}
    <a class="dropdown-item" href="{%  url 'admin:index' %}">Ад-
минПанель</a>
            {% endif %}
    <a class="dropdown-item" href="{%  url 'main:profile' %}">Мои
публикации</a>
    <a class="dropdown-item" href="{% url 'main:profile_change'
%}">Изменить личные данные</a>
    <a class="dropdown-item" href="{% url 'main:password_change'
%}">Изменить пароль</a>
    <a class="dropdown-item" href="{% url 'main:profile_delete'
%}">Удалить</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="{% url 'main:logout'
%}">Выйти</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="#">Удалить</a>
    </div>
    </li>
       {% else %}
    <li class="nav-item"><a class="nav-link"
                href="{% url 'main:login' %}">Вход</a></li>
                {% endif %}

    <li class="nav-item">
    <a class="nav-link text-light  px-3" href="{% url
'main:register' %}">Регистрация</a>
```

```
      </li>
      </ul>

      </nav>
      <div class="row">
      <nav class="col-md-auto nav flex-column border pl-4">
      <a class="nav-link root pt-15" href="{% url 'main:index'
%}">Главная</a>
      <span class="nav-link root font-weight-bold">Выделенный заго-
ловок</span>
                      {% for rubric in rubrics %}
                         {% ifchanged rubric.super_rubric.pk %}
      <span class="nav-link root font-weight-bold">
                              {{ rubric.super_rubric.name }}</span>
                          {% endifchanged %}
      <a class="nav-link" href="{% url 'main:by_rubric'
pk=rubric.pk %}">
                                              {{ rubric.name }}</a>
                                              {% endfor %}
      <a class="nav-link root pt-5" href="{% url 'main:other'
page='about' %}">Осайте</a>
      </nav>

      <section class="col border py-2 ">

                                     {% bootstrap_messages %}
                                     {% block content %}
      {% endblock %}
      </section>
      </div>
      <footer class="mt-3">
      <p class="text-center font-italic">&copy; Екатеринбург.
УрГПУ. ИСИТ-1931z. 2024.</p>
      </footer>
```

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

Исходный код шаблона **index.html** (главная страница).

```
{% extends "layout/basic.html" %}

{% load thumbnail %}
{% load static %}

{% block content %}
<h2>Последние 10 публикаций</h2>
{% if bbs %}
<ul class="list-unstyled">
    {% for bb in bbs %}
<li class="media my-5 p-3 border">
        {% url 'main:detail' rubric_pk=bb.rubric.pk pk=bb.pk
as url %}
<a href="{{ url }}{{ all }}">
        {% if bb.image %}
<img class="mr-3" src="{% thumbnail bb.image 'default' %}">
        {% else %}
<img class="mr-3" src="{% static 'main/empty.jpg' %}">
        {% endif %}
</a>
<div class="media-body">
<h3><a href="{{ url }}{{ all }}">
          {{ bb.title }}</a></h3>
<div>{{ bb.content }}</div>

<p class="text-right font-italic">{{ bb.created_at }}</p>
</div>
</li>
    {% endfor %}
```

```
</ul>
{% endif %}
```

Исходный код шаблона **change_user_info.html** (изменение пользовательской информации).

```
{% extends "layout/basic.html" %}
{% load bootstrap4 %}
{% block title %}Правка личных данных{% endblock %}
{% block content %}
<h2>Правка личных данных пользователя{{user.username}}</h2>
<form method="post">
    {% csrf_token %}
    {% bootstrap_form form layout='horizontal' %}
    {% buttons submit='Сохранить' %}{% endbuttons %}
</form>
{% endblock %}
```

## Приложение 2. Исходный код модуля Urls.py (Маршруты)

```
from django.urls import path
from .views import index, other_page, BBLoginView, profile, BBLogoutView, \
    ChangeUserInfoView, BBPasswordChangeView, RegisterUserView, \
    RegisterDoneView, user_activate, DeleteUserView, BBPasswordResetView, \
    BBPasswordResetDoneView, BBPasswordResetConfirmView, \
    BBPasswordResetCompleteView, by_rubric, detail, profile_bb_detail, \
    profile_bb_add, profile_bb_change, profile_bb_delete, show_resume, show_vacancy
```

```
app_name = 'main'
urlpatterns = [
    path('accounts/register/activate/<str:sign>/', user_activate,
name='register_activate'),
    path('accounts/register/done/', RegisterDoneView.as_view(),
                                    name='register_done'),
    path('accounts/register/', RegisterUser-
View.as_view(),name='register'),
    path('accounts/logout/', BBLogoutView.as_view(),
name='logout'),
    path('accounts/password/change/', BBPasswordChange-
View.as_view(),
                                     name='password_change'),
    path('accounts/profile/delete/', DeleteUserView.as_view(),
                                     name='profile_delete'),
    path('accounts/profile/change/', ChangeUserInfoView.as_view(),
                                     name='profile_change'),
    path('accounts/profile/change/<int:pk>/', profile_bb_change,
name='profile_bb_change'),
    path('accounts/profile/delete/<int:pk>/', profile_bb_delete,
name='profile_bb_delete'),
    path('accounts/profile/add/', profile_bb_add,
name='profile_bb_add'),
    path('accounts/profile/<int:pk>/', profile_bb_detail,
                                      name='profile_bb_detail'),
    path('accounts/profile/', profile, name='profile'),
    path('accounts/password/reset/done/', BBPasswordResetDone-
View.as_view(),
name='password_reset_done'),
```

```
    path('accounts/password/reset/', BBPasswordReset-
View.as_view(),
                                      name='password_reset'),
    path('accounts/password/confirm/complete/',
                                     BBPasswordResetComplete-
View.as_view(),

name='password_reset_complete'),
    path('accounts/password/confirm/<uidb64>/<token>/',
                                     BBPasswordResetConfirm-
View.as_view(),

name='password_reset_confirm'),
    path('accounts/login/', BBLoginView.as_view(), name='login'),
    path('<int:rubric_pk>/<int:pk>/', detail, name='detail'),
    path('<int:pk>/', by_rubric, name='by_rubric'),
    path('vacancy/', show_vacancy, name='show_vacancy'),
    path('resume/', show_resume, name='show_resume'),
    path('<str:page>/', other_page, name='other'),
    path('', index, name='index'),
```

## Приложение 3.Исходный код модуля Views.py

```
from django.shortcuts import render, get_object_or_404, redirect
from django.http import HttpResponse, Http404 #  HttpResponse по-
зволяет отправить текстовое содержимое
from django.template import TemplateDoesNotExist
from django.template.loader import get_template
from django.contrib.auth.views import LoginView, LogoutView, \
    PasswordChangeView, PasswordResetView, PasswordResetDoneView,
\
    PasswordResetConfirmView, PasswordResetCompleteView
```

```
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import UpdateView, CreateView, \
    DeleteView
from django.contrib.messages.views import SuccessMessageMixin
from django.urls import reverse_lazy
from django.views.generic.base import TemplateView
from django.core.signing import BadSignature
from django.contrib.auth import logout
from django.contrib import messages
from django.core.paginator import Paginator
from django.db.models import Q
from .models import AdvUser, SubRubric, Bb, Comment
from .forms import ChangeUserInfoForm, RegisterUserForm, Sear-
chForm, \
 UserCommentForm, GuestCommentForm,BbForm , AIFormSet
from .utilities import signer


# Контроллер для главной страницы
def index(request):
bbs = Bb.objects.filter(is_active=True)[:10]
    context = {'bbs': bbs}
    return render(request, 'main/index.html', context)



# Контроллер для отображения всех вакансий
def show_vacancy(request):
vacancy = Bb.objects.filter(is_active=True, ac-
count_adds_vacancy=True)
  context = {'vacancy':vacancy}
  return render (request, 'main/show_vacancy.html', context)
```

```
# Контроллер для отображения всех резюме
def show_resume(request):
resume = Bb.objects.filter(is_active=True, ac-
count_adds_resume=True)
    context = {'resume':resume}
    return render (request, 'main/show_resume.html', context)



# Контроллер для вспомогательных страниц
def other_page(request, page):
try:
        template = get_template('main/' + page + '.html')
    except TemplateDoesNotExist:
        raise Http404
    return HttpResponse(template.render(request=request))

# Контроллер класса "Входа"
class BBLoginView(LoginView):
    template_name = 'main/login.html'



@login_required  # Декоратор который проверяет залогинился ли
пользователь
def profile(request):  # Контроллер страницы пользовательского
профиля
    bbs = Bb.objects.filter(author=request.user.pk) # Фильтрация
публикаций по значению поля author
    context = {'bbs': bbs}  # (Ключ автора объявления, которым яв-
ляется зарегистрированный пользователь), сравнивая
     # это значение с ключом текущего пользователя
```

```
return render(request, 'main/profile.html', context)


 # Контроллеркласса "Выхода"
class BBLogoutView(LoginRequiredMixin, LogoutView):
    template_name = 'main/logout.html'


 # Контролер класса изменяющий данные пользователя
class ChangeUserInfoView(SuccessMessageMixin, LoginRequiredMixin,
                                                    UpdateView):
    model = AdvUser
    template_name = 'main/change_user_info.html'
    form_class = ChangeUserInfoForm
    success_url = reverse_lazy('main:profile')
success_message = 'Данные пользователя изменены'
# В процессе работы контроллер должен извлечь из модели AdvUser
запись представляющую текущего пользователя
def setup(self, request, *args, **kwargs):
self.user_id = request.user.pk  # Получаем ключ текущего пользова-
теля (сохраняем его в атрибуте user_id)
return super().setup(request, *args, **kwargs)

    def get_object(self, queryset=None):
        if not queryset:
            queryset = self.get_queryset()
        return get_object_or_404(queryset, pk=self.user_id)


 # Контроллер класса изменяющий пароль пользователя
```

```
class BBPasswordChangeView(SuccessMessageMixin, LoginRequiredMix-
in,
                                                    PasswordChange-
View):
    template_name = 'main/password_change.html'
    success_url = reverse_lazy('main:profile')
success_message = 'Пароль пользователя изменен'


 # Контроллер класса регистрирующий пользователя
class RegisterUserView(CreateView):
    model = AdvUser


    template_name = 'main/register_user.html'
    form_class = RegisterUserForm
    success_url = reverse_lazy('main:register_done')


# Контроллер класс выводящий сообщение об успешной регистрации
class RegisterDoneView(TemplateView):
    template_name = 'main/register_done.html'


# Контроллер для активации пользователя
def user_activate(request, sign):
try:
        username = signer.unsign(sign)
    except BadSignature:
        return render(request, 'main/bad_signature.html')
    user = get_object_or_404(AdvUser, username=username)
```

```
    if user.is_activated:
        template = 'main/user_is_activated.html'
    else:
        template = 'main/activation_done.html'
        user.is_active = True
        user.is_activated = True
        user.save()
    return render(request, template)



 # Контроллер класса для удаления пользовтеля
class DeleteUserView(LoginRequiredMixin, DeleteView):
    model = AdvUser
    template_name = 'main/delete_user.html'
    success_url = reverse_lazy('main:index')

    def setup(self, request, *args, **kwargs):
        self.user_id = request.user.pk
        return super().setup(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        logout(request)
        messages.add_message(request, messages.SUCCESS,
'Пользователь удален')
        return super().post(request, *args, **kwargs)

    def get_object(self, queryset=None):
        if not queryset:
            queryset = self.get_queryset()
        return get_object_or_404(queryset, pk=self.user_id)
```

```
class BBPasswordResetView(PasswordResetView):
    template_name = 'main/password_reset.html'
    subject_template_name = 'email/reset_letter_subject.txt'
    email_template_name = 'email/reset_letter_body.txt'
    success_url = reverse_lazy('main:password_reset_done')

class BBPasswordResetDoneView(PasswordResetDoneView):
    template_name = 'main/password_reset_done.html'

class BBPasswordResetConfirmView(PasswordResetConfirmView):
    template_name = 'main/password_confirm.html'
    success_url = reverse_lazy('main:password_reset_complete')

class BBPasswordResetCompleteView(PasswordResetCompleteView):
    template_name = 'main/password_complete.html'

def by_rubric(request, pk):
    rubric = get_object_or_404(SubRubric, pk=pk)
    bbs = Bb.objects.filter(is_active=True, rubric=pk)
    if 'keyword' in request.GET:
        keyword = request.GET['keyword']
        q = Q(title__icontains=keyword) |
Q(content__icontains=keyword)
        bbs = bbs.filter(q)
    else:
        keyword = ''
    form = SearchForm(initial={'keyword': keyword})
    paginator = Paginator(bbs, 2)
```

```
    if 'page' in request.GET:
        page_num = request.GET['page']
    else:
        page_num = 1
    page = paginator.get_page(page_num)
    context = {'rubric': rubric, 'page': page, 'bbs':
page.object_list,
               'form': form}
    return render(request, 'main/by_rubric.html', context)


def detail(request, rubric_pk, pk):
    bb = Bb.objects.get(pk=pk)
    ais = bb.additionalimage_set.all()
    comments = Comment.objects.filter(bb=pk, is_active=True)
    initial = {'bb': bb.pk}
    if request.user.is_authenticated:
        initial['author'] = request.user.username
        form_class = UserCommentForm
    else:
        form_class = GuestCommentForm
    form = form_class(initial=initial)
    if request.method == 'POST':
        c_form = form_class(request.POST)
        if c_form.is_valid():
            c_form.save()
            messages.add_message(request, messages.SUCCESS,
'Комментарий добавлен')
        else:
            form = c_form
messages.add_message(request, messages.WARNING,
                                 'Комментарий не добавлен')
```

```
    context = {'bb': bb, 'ais': ais, 'comments': comments, 'form':
form}
    return render(request, 'main/detail.html', context)


@login_required # только зарегистрированным пользователям
def profile_bb_detail(request, pk):  # вывод страницы сведений о
публикации
bb = get_object_or_404(Bb, pk=pk)
    ais = bb.additionalimage_set.all()
    comments = Comment.objects.filter(bb=pk, is_active=True)
    context = {'bb': bb, 'ais': ais, 'comments': comments}
    return render(request, 'main/profile_bb_detail.html', context)



# ДОБАВЛЕНИЕ ПУБЛИКАЦИИ
@login_required  # только зарегистрированным пользователям
def profile_bb_add(request): # Добавление публикации



   if request.method == 'POST':
     form = BbForm(request.POST, request.FILES)
     if form.is_valid():


     bb = form.save()
     formset = AIFormSet(request.POST, request.FILES, instance=bb)
     if formset.is_valid():
            formset.save()
            messages.add_message(request, messages.SUCCESS,
                                 'Объявление добавлено')
     return redirect('main:profile')
   else:
```

```
        # Автоматизируем вставку в поля значений по умолчанию
form = BbForm(initial={'author': request.user.pk,
                               'email': request.user.email
                               ,
                               'account_adds_vacancy': re-
quest.user.account_add_vacancy,
                               'account_adds_resume': re-
quest.user.account_add_resume })
    formset = AIFormSet()
    context = {'form': form, 'formset': formset}
    return render(request, 'main/profile_bb_add.html', context)


@login_required  #  только зарегистрированным пользователям
defprofile_bb_change(request, pk):  # Исправление публикации
bb = get_object_or_404(Bb, pk=pk)
    if request.method == 'POST':
        form = BbForm(request.POST, request.FILES, instance=bb)
        if form.is_valid():
            bb = form.save()
            formset = AIFormSet(request.POST, request.FILES, in-
stance=bb)
            if formset.is_valid():
                formset.save()
                messages.add_message(request, messages.SUCCESS,
                                     'Объявление исправлено')
                return redirect('main:profile')
    else:
         form = BbForm(instance=bb)
         formset = AIFormSet(instance=bb)
    context = {'form': form, 'formset': formset}
```

```
    return render(request, 'main/profile_bb_change.html', context)


@login_required  # только зарегистрированным пользователям
defprofile_bb_delete(request, pk): # Удаление публикации
bb = get_object_or_404(Bb, pk=pk)
    if request.method == 'POST':
        bb.delete()
        messages.add_message(request, messages.SUCCESS,
'Объявлениеудалено')
        return redirect('main:profile')
    else:
        context = {'bb': bb}
        return render(request, 'main/profile_bb_delete.html', con-
text)
```

## Приложение 4.Исходный код модуля Models.py (Модели)

```
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.db.models.signals import post_save
from .utilities import get_timestamp_path,
send_new_comment_notification


# Модельпользователя
class AdvUser(AbstractUser):
    is_activated = models.BooleanField(default=True,
db_index=True, verbose_name='Прошел активацию?')
    account_add_vacancy = models.BooleanField(default=False,
verbose_name='Аккаунт для добавления вакансии (для работодате-
лей)')
account_add_resume = models.BooleanField(default=False,
```

```
verbose_name='Аккаунт для добавления резюме (поиск работы)')
send_messages = models.BooleanField(default=True,
verbose_name='Слать оповещения о новых комментариях?')

# При удалении пользователя удаляются оставленные им обявления
def delete(self, *args, **kwargs):
        for bb in self.bb_set.all():
            bb.delete()
        super().delete(*args, **kwargs)

    class Meta(AbstractUser.Meta):
        pass

# Класс модели Rubric в которой хранятся надрубрики и подрубрики
class Rubric(models.Model):
    name = models.CharField(max_length=20, db_index=True,
unique=True,
                            verbose_name='Название')
    order = models.SmallIntegerField(default=0, db_index=True,
                                     verbose_name='Порядок')
    super_rubric = models.ForeignKey('SuperRubric',
on_delete=models.PROTECT,
                     null=True, blank=True, ver-
bose_name='Надрубрика')

# Диспетчер записей SuperRubric
class SuperRubricManager(models.Manager):
    def get_queryset(self):
        return su-
per().get_queryset().filter(super_rubric__isnull=True)

# Класс Модели SuperRubric
```

```
class SuperRubric(Rubric):
    objects = SuperRubricManager()

    def __str__(self):
        return self.name

    class Meta:
        proxy = True
        ordering = ('order', 'name')
        verbose_name = 'Надрубрика'
        verbose_name_plural = 'Надрубрики'

# Диспетчер записей SubRubric (Подрубрики)
class SubRubricManager(models.Manager):
    def get_queryset(self):
       return su-
per().get_queryset().filter(super_rubric__isnull=False)

# Модель подрубрик SubRubric
class SubRubric(Rubric):

    SuperRubricManager
    objects = SubRubricManager()



    def __str__(self):
         return '%s - %s' % (self.super_rubric.name, self.name)



    class Meta:
        proxy = True
```

```
        ordering = ('super_rubric__order', 'super_rubric__name',
'order',
                    'name')
        verbose_name = 'Подрубрика'
        verbose_name_plural = 'Подрубрики'


# Модель хранящая публикации
class Bb(models.Model):

    SCHEDULE_CHOICE = (
    ('Полная занятость', 'Полная занятость'),
    ('Гибкий график', 'Гибкий график'),
    ('Сменный график', 'Сменный график'),
    ('Скользящий график', 'Скользящий график'),
    ('Разрывной график', 'Разрывной график'),
    ('Вахтовый метод', 'Вахтовый метод'),
    ('Другое', 'Другое'),
    )


    EDUCATION_CHOICE = (
    ('Среднее', 'Среднее'),
    ('Среднее-профессиональное (СПО)', 'Среднее-профессиональное
(СПО)'),
    ('Высшее (ВО)', 'Высшее (ВО)'),
)


    rubric = models.ForeignKey(SubRubric,
on_delete=models.PROTECT,verbose_name='Категория')
    title = models.CharField(max_length=100,
verbose_name='Должность')
```

```
    content = models.TextField(verbose_name='Описание')  #  (Опи-
сание) Овакансии / Осебе

    image = models.ImageField(blank=True, upl-
oad_to=get_timestamp_path,
                                verbose_name='Изображение')


    schedule = models.CharField(max_length=40, choices = SCHE-
DULE_CHOICE, default='Полнаязанятость',
verbose_name='Графикработы')

    experience = models.CharField(max_length=100,
default='Неуказан', verbose_name='Опытработы')

    education = models.CharField(max_length=40, default='Среднее-
профессиональное', choices = EDUCATION_CHOICE,
verbose_name='Образование')

    empoloyment_area = models.CharField(max_length=150,
default='Неуказан', verbose_name='Районтрудоустройства')

    salary_from = models.IntegerField(default='0',
verbose_name='Зарплатаот')

    salary_up_to = models.IntegerField(default='0',
verbose_name='Зарплатадо')

    telephone = models.CharField(max_length=40, de-
fault='Неуказан', verbose_name='Телефон')

    name_contact = models.CharField(max_length=100,
default='Неуказано', verbose_name='ФИОсоискателя')

    organization = models.CharField(max_length=100,
default='Неуказано', verbose_name='Названиеорганизации')

    email = models.EmailField(default=AdvUser.objects.filter()
,verbose_name='E-mail')

    author = models.ForeignKey(AdvUser, on_delete=models.CASCADE,
                                verbose_name='Авторпубликации')

    is_active = models.BooleanField(default=True, db_index=True,
verbose_name='Выводить в списке?')

created_at = models.DateTimeField(auto_now_add=True,
db_index=True,
                                    verbose_name='Опубликовано')

    account_adds_vacancy = models.BooleanField(default=False,)
```

```
    account_adds_resume = models.BooleanField(default=False,)


# Функция для вывода анонса публикации. В данном случае это 180
символов. Необходимо вызвать эту
     # функцию в целевом шаблоне вместо имени поля content
def get_summary(self):
        return self.content[:180]



    def delete(self, *args, **kwargs):
        for ai in self.additionalimage_set.all():
            ai.delete()
        super().delete(*args, **kwargs)

    class Meta:
        verbose_name_plural = 'Публикации'
verbose_name = 'Публикация'
        ordering = ['-created_at'] # Сортировка по полю (-) меняет
порядок сортировки



# Модель дополнительных иллюстраций в объявлении
class AdditionalImage(models.Model):
bb = models.ForeignKey(Bb, on_delete=models.CASCADE,
                               verbose_name='Объявление')
    image = models.ImageField(upload_to=get_timestamp_path,
                                  verbose_name='Изображение')

    class Meta:
        verbose_name_plural = 'Дополнительные иллюстрации'
        verbose_name = 'Дополнительная иллюстрация'
```

```
# Модель комментариев
class Comment(models.Model):
    bb = models.ForeignKey(Bb, on_delete=models.CASCADE,
                                verbose_name='Объявление')
    author = models.CharField(max_length=30, verbose_name='Автор')
    content = models.TextField(verbose_name='Содержание')
    is_active = models.BooleanField(default=True, db_index=True,
verbose_name='Выводить на экран?')
created_at = models.DateTimeField(auto_now_add=True,
db_index=True,
                                        verbose_name='Опубликован')

    class Meta:
        verbose_name_plural = 'Комментарии'
        verbose_name = 'Комментарий'
        ordering = ['created_at']


def post_save_dispatcher(sender, **kwargs):
    author = kwargs['instance'].bb.author
    if kwargs['created'] and author.send_messages:
        send_new_comment_notification(kwargs['instance'])

post_save.connect(post_save_dispatcher, sender=Comment)
```

## Приложение 5.Исходный код модуля Forms.py (Формы)

```
from django import forms
```

```
from django.contrib.auth import password_validation

from django.core.exceptions import ValidationError

from django.forms import inlineformset_factory

from captcha.fields import CaptchaField

from .models import AdvUser, SuperRubric, SubRubric, Bb, Additio-
nalImage, \

    Comment

from .apps import user_registered



# Форма для ввода основных данных

class ChangeUserInfoForm(forms.ModelForm):

    email = forms.EmailField(required=True, label='Адрес электрон-
ной почты')


# Вложеный класс Meta внутри которого указываем характеристики для
нашего целевого класса ChangeUserInfoForm

    class Meta:

        model = AdvUser # Указываем модель с которой работаем

        fields = ('username', 'email', 'first_name',
'last_name','send_messages') # Какие поля должны быть выведены
внутри формы


# Форма для занесения сведений о новом пользователе

class RegisterUserForm(forms.ModelForm):

    email = forms.EmailField(required=True,
label='Адресэлектроннойпочты')

    password1 = forms.CharField(label='Пароль', wid-
get=forms.PasswordInput,


help_text=password_validation.password_validators_help_text_html()
)

    password2 = forms.CharField(label='Пароль (повторно)',
```

```
widget=forms.PasswordInput,
      help_text='Введите тот же самый пароль еще раз для
проверки')

account_add_vacancy = forms.BooleanField(
label='Аккаунт для добавления вакансии (для работодателей)')
account_add_resume = forms.BooleanField(
label='Аккаунт для добавления резюме (поиск работы)')



def clean_password1(self):
        password1 = self.cleaned_data['password1']
        if password1:
            password_validation.validate_password(password1)
        return password1

    def clean(self):
        super().clean()
        password1 = self.cleaned_data['password1']
        password2 = self.cleaned_data['password2']
        if password1 and password2 and password1 != password2:
            errors = {'password2': ValidationError(
              'Введенныепаролинесовпадают',
code='password_mismatch')}
            raise ValidationError(errors)

    def save(self, commit=True):
        user = super().save(commit=False)
        user.set_password(self.cleaned_data['password1'])
user.is_active = False # Является ли пользователь активным по
умолчанию False
```

```
        user.is_activated = False # Выполнил ли пользователь процедуру активации по умолчанию False
        user.group = 'Пользователь'

        if commit:
            user.save()
        user_registered.send(RegisterUserForm, instance=user)
        return user

    class Meta:
        model = AdvUser
        fields = (
                  'username', 'email', 'password1', 'password2',
                  'first_name', 'last_name', 'send_messages')

# Форма Рубрик (делаем поле "Надрубрики" SuperRubric обязательным)
class SubRubricForm(forms.ModelForm):
    super_rubric = forms.ModelChoiceField(queryset=SuperRubric.objects.all(),
                                          empty_label=None, label='Надрубрика',
                                          required=True)

    class Meta:
        model = SubRubric
fields = '__all__'


# Форма поиска
class SearchForm(forms.Form):
    keyword = forms.CharField(required=False, max_length=20, label='')
```

```
# ФОРМА ДЛЯ ДОБАВЛЕНИЯ ПУБЛИКАЦИИ
class BbForm(forms.ModelForm):


class Meta:
            model = Bb
            #fields = '__all__'
            fields =
('author','rubric','title','organization','content','image','sched
ule','experience','education',

'empoloyment_area','salary_from','salary_up_to','telephone','email
',

'name_contact','account_adds_vacancy','account_adds_resume' )


            labels = {'author' : 'Имяпользователя','rubric':
'Категория ','title': 'Должность ','content': 'Овакансии/ Осебе
','image': 'Изображение ',
                'schedule': 'Графикработы ','experience':
'Опытработы ','education': 'Образование ',
                'empoloyment_area': 'Районтрудоустройства
','salary_from': 'Зарплатаот ','salary_up_to': 'Зарплатадо',
'telephone': 'Телефон ','email': 'Электронная почта',
'name_contact': 'Контактноелицо','organization': 'Организация',
                'account_adds_vacancy':
'Вакансия','account_adds_resume': 'Резюме'
}


            help_texts = {'rubric': 'Укажите категорию в которой
хотите разместить Вашу публикацию. ',
            'title': 'Укажите Должность) ',
            'author' : '',
```

```
            'content': 'Расскажите, в зависимости от ситуации о
вакансии или о себе. ',

            'image': 'Добавьте изображение (Не обязательно) ',

            'schedule': 'Выберите из списка график работы. ',

            'experience': 'выберите из списка оптимальное значение
подходящее для Вашего конкретного случая.  ',

            'education': 'Выберите из списка соответствующий уро-
вень образования . ',

            'empoloyment_area': 'Укажите регион или адрес местона-
хождения вакансии. ',

            'salary_from': 'Укажите минимальный уровень заработной
платы. ',

            'salary_up_to': 'Укажите максимальный уровень зарплаты
(Если такой предел имеется). Иначе оставьте это поле пустым.  ',

            'telephone': 'Укажите телефон для связи. ',

            'email': 'Укажите адрес электронной почты для связи.
',

            'name_contact': 'Укажите как к Вам можно обращаться
(Фамилия, Имя и т.п..)',

            'organization': 'Если вы публикуете вакансию.Укажите
название своей организации. При публикации Резюме это поле запол-
нять не нужно ',

            'account_adds_vacancy': '',

            'account_adds_resume': ''


    }

           widgets = {'author': forms.HiddenInput  # Делаем поле
скрытым (значение передаётся автоматически)

}


AIFormSet = inlineformset_factory(Bb, AdditionalImage,
fields='__all__')
```

```
# Форма комментариев оставленных зарегистрированными пользователя-
ми
class UserCommentForm(forms.ModelForm):
class Meta:
        model = Comment
        exclude = ('is_active',)
widgets = {'bb': forms.HiddenInput}


# Форма Комментариев оставленных Незарегистрированными пользовате-
лями сайта (Гостями)
class GuestCommentForm(forms.ModelForm):
    captcha = CaptchaField(label='Введитетекстскартинки',
              error_messages={'invalid': 'Неправильныйтекст'})

    class Meta:
        model = Comment
        exclude = ('is_active',)
        widgets = {'bb': forms.HiddenInput}
```

## Приложение 6.Admin.py (Административнаяпанель)

```
from django.contrib import admin
import datetime
from .models import AdvUser, Bb, AdditionalImage, Comment
from .forms import SubRubricForm
from .utilities import send_activation_notification

def send_activation_notifications(modeladmin, request, queryset):
    for rec in queryset:
        if not rec.is_activated:
```

```
            send_activation_notification(rec)
    modeladmin.message_user(request, 'Письма с требованиями
отправлены')
send_activation_notifications.short_description = \
'Отправка писем с требованиями активации'


class NonactivatedFilter(admin.SimpleListFilter):
    title = 'Прошли активацию?'
    parameter_name = 'actstate'


    def lookups(self, request, model_admin):
return (
                    ('activated', 'Прошли'),
                    ('threedays', 'Не прошли более 3 дней'),
                    ('week', 'Не прошли более недели'),
)


    def queryset(self, request, queryset):
        val = self.value()
        if val == 'activated':
            return queryset.filter(is_active=True,
is_activated=True)
        elif val == 'threedays':
            d = datetime.date.today() - datetime.timedelta(days=3)
            return queryset.filter(is_active=False,
is_activated=False,
                                   date_joined__date__lt=d)
        elif val == 'week':
            d = datetime.date.today() - date-
time.timedelta(weeks=1)
            return queryset.filter(is_active=False,
is_activated=False,
```

```
                                    date_joined__date__lt=d)


class AdvUserAdmin(admin.ModelAdmin):
    list_display = ('__str__', 'is_activated', 'date_joined')
    search_fields = ('username', 'email', 'first_name',
'last_name')
    list_filter = (NonactivatedFilter,)
    fields = (('username', 'email'), ('first_name', 'last_name'),
              ('send_messages', 'is_active', 'is_activated'),
              ('account_add_vacancy', 'account_add_resume'),
              ('is_staff', 'is_superuser'),
              'groups', 'user_permissions',
              ('last_login', 'date_joined'))
    readonly_fields = ('last_login', 'date_joined')
    actions = (send_activation_notifications,)

admin.site.register(AdvUser, AdvUserAdmin)

from .models import SuperRubric, SubRubric

# Встроенный редактор надрубрик (заполнение надрубрики подрубрика-
ми)
class SubRubricInline(admin.TabularInline):
    model = SubRubric

 # Класс редактора надрубрик
class SuperRubricAdmin(admin.ModelAdmin):
    exclude = ('super_rubric',)
    inlines = (SubRubricInline,)

admin.site.register(SuperRubric, SuperRubricAdmin)
```

```
# Класс редактора подрубрик
class SubRubricAdmin(admin.ModelAdmin):
    form = SubRubricForm

admin.site.register(SubRubric, SubRubricAdmin)


# Встроенный редактор дополнительных иллюстраций
class AdditionalImageInline(admin.TabularInline):
    model = AdditionalImage

# Редактор объявлений в АдминПанели (+)
class BbAdmin(admin.ModelAdmin):
    list_display = ('rubric', 'title', 'content', 'author',
'created_at')
    fields = (('rubric', 'author'), 'title', 'content',
'image','schedule','education','experience','empoloyment_area',

'salary_from','salary_up_to','telephone','email','is_active' )
    inlines = (AdditionalImageInline,)

admin.site.register(Bb, BbAdmin)

# Редактор для модели Comment
class CommentAdmin(admin.ModelAdmin):
    list_display = ('author', 'content', 'created_at',
'is_active')
    list_display_links = ('author', 'content')
    list_filter = ('is_active',)
    search_fields = ('author', 'content',)
    date_hierarchy = 'created_at'
```

```
    fields = ('author', 'content', 'is_active', 'created_at')
    readonly_fields = ('created_at',)

admin.site.register(Comment, CommentAdmin)
```