



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные  
технологии»

## РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

### *К КУРСОВОМУ ПРОЕКТУ*

#### *НА ТЕМУ:*

*«Разработка онлайн-платформы для поиска  
вакансий»*

Студент ИУ7-65Б  
(Группа)

Кондрашова О.П.  
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта

Гаврилова Ю.М.  
(Подпись, дата) (И.О.Фамилия)

2020 г.

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

УТВЕРЖДАЮ  
Заведующий кафедрой \_\_\_\_\_ ИУ7  
(Индекс)  
\_\_\_\_\_ И. В. Рудаков  
(И.О.Фамилия)  
« \_\_\_\_\_ » \_\_\_\_\_ 2020 г.

**З А Д А Н И Е  
на выполнение курсового проекта**

по дисциплине \_\_\_\_\_ Базы данных \_\_\_\_\_

Студент группы \_\_\_\_\_ ИУ7-65Б \_\_\_\_\_

\_\_\_\_\_ Кондрашова Ольга Павловна  
(Фамилия, имя, отчество)

Тема курсового проекта \_\_\_\_\_ Разработка онлайн-платформы для поиска вакансий \_\_\_\_\_

Направленность КП (учебный, исследовательский, практический, производственный, др.)  
\_\_\_\_\_ Учебный \_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_ Кафедра \_\_\_\_\_

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

**Задание** Разработать клиент-серверное приложение для поиска вакансий. Спроектировать и реализовать базу данных приложения. Реализовать приложения для взаимодействия с базой данных. Реализовать авторизацию и доступ к списку вакансий с возможностью подать заявку на конкретную вакансию, а также просмотр выбранных пользователем вакансий.

**Оформление курсового проекта:**

Расчетно-пояснительная записка на 20-30 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

На защиту проекта должна быть представлена презентация, состоящая из 10-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО.

Дата выдачи задания « 19 » марта 2020 г.

**Руководитель курсового проекта**

**Студент**

_____	_____ Гаврилова Ю.М.
(Подпись, дата)	(И.О.Фамилия)
_____	_____ Кондрашова О.П.
(Подпись, дата)	(И.О.Фамилия)

# Содержание

Введение.....	4
1. Аналитический раздел .....	5
1.1 Формализация задачи.....	5
1.2 Реляционные базы данных.....	5
1.2.1 Структурная часть реляционной модели .....	6
1.2.2 Целостная часть реляционной модели .....	7
1.2.3 Манипуляционная часть реляционной модели.....	7
1.3 СУБД.....	8
Вывод .....	8
2. Конструкторский раздел.....	9
2.1 Диаграмма вариантов использования .....	9
2.2 ER диаграмма .....	10
2.3 Проектирование таблиц базы данных .....	10
2.4 Паттерн модель-представление-контроллер.....	13
2.4.1 Модели.....	13
2.4.2 Представления .....	14
2.4.3 Контроллер .....	14
2.5 Регистрация и аутентификация пользователей .....	15
Вывод .....	15
3. Технологический раздел.....	16
3.1 Выбор технологического стека.....	16
3.2 Реализация хранения данных.....	18
3.3 Реализация доступа к данным.....	20
3.4 Frontend-разработка .....	21
3.5 Интерфейс приложения.....	22
Вывод .....	25
Заключение .....	26
Список литературы.....	27

## Введение

Ежедневно тысячи людей по всей стране занимаются поиском работы. Раньше процесс поиска был долгим и затруднительным: приходилась искать работу через объявления или знакомых, часами обзванивать компании или приезжать к ним в офис. С развитием информационных технологий процесс поиска подходящей вакансии стал существенно проще. Приобрели популярность специализированные сайты вакансий, где пользователь может найти подходящую ему вакансию по заданным фильтрам и моментально отправить заявку на нее.

Целью данной курсовой работы является разработка онлайн-платформы для поиска вакансий.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать существующие СУБД;
- спроектировать базу данных для хранения и структурирования данных;
- реализовать приложение для взаимодействия с базой данных.

# **1. Аналитический раздел**

В данном разделе выполняется постановка задачи, проводится анализ существующих СУБД и технологий.

## **1.1 Формализация задачи**

В ходе выполнения данной курсовой работы должно быть спроектировано и реализовано клиент-серверное приложение с поддержкой следующего функционала:

- предоставляет доступ к списку всех существующих вакансий;
- просмотр информации о конкретной вакансии;
- возможность подать заявку на выбранную вакансию;
- поиск вакансий по нескольким параметрам: название, наименование индустрии, компания;
- регистрация новых пользователей;
- авторизация уже зарегистрированных пользователей.

## **1.2 Реляционные базы данных**

База данных представляет собой совокупность определенным образом организованных данных, которые хранятся в памяти вычислительной системы и отображают состояние объектов и их взаимосвязи в рассматриваемой предметной области.

Реляционная база данных – это набор отношений, имена которых совпадают с именами схем отношений в схеме базы данных.

Реляционная модель данных включает следующие компоненты:

- структурный (данные в базе данных представляют собой набор отношений);
- целостностный (отношения (таблицы) отвечают определенным условиям целостности);

- манипуляционный (манипулирования отношениями осуществляется средствами реляционной алгебры и/или реляционного исчисления). Кроме того, в состав реляционной модели данных включают теорию нормализации.

### 1.2.1 Структурная часть реляционной модели

Структурная часть реляционной модели описывает, из каких объектов состоит реляционная модель. Основной структурой данных, используемой в реляционной модели, являются нормализованные «n-арные» отношения.

Домен можно рассматривать как подмножество значений некоторого типа данных, имеющих определенный смысл. Домен характеризуется следующими свойствами:

- домен имеет уникальное имя (в пределах базы данных);
- домен определен на некотором типе данных или на другом домене;
- домен может иметь некоторое логическое условие, позволяющее описать подмножество данных, допустимых для данного домена;
- домен несет определенную смысловую нагрузку.

Атрибут отношения – это пара вида <имя\_атрибута, имя\_домена >. Имена атрибутов должны быть уникальны в пределах отношения. Часто имена атрибутов отношения совпадают с именами соответствующих доменов.

Схема отношения – это именованное множество упорядоченных пар <имя\_атрибута, имя\_домена>. Степенью или «арностью» схемы отношения является мощность этого множества. Схема базы данных в реляционной модели – это множество именованных схем отношений.

Кортеж, соответствующий данной схеме отношения, – это множество упорядоченных пар <имя\_атрибута, значение\_атрибута>, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. Значение атрибута должно быть допустимым значением домена, на котором определен данный атрибут. Степень или «арность» кортежа совпадает с «арностью» соответствующей схемы отношения.

## **1.2.2 Целостная часть реляционной модели**

В целостностной части реляционной модели фиксируются два базовых требования целостности, которые должны выполняться для любых отношений в любых реляционных базах данных. Это целостность сущностей и ссылочная целостность (или целостность внешних ключей).

Требование целостности сущностей заключается в следующем: каждый кортеж любого отношения должен отличаться от любого другого кортежа этого отношения (т.е. любое отношение должно обладать потенциальным ключом). Поддержание целостности сущностей обеспечивается средствами СУБД. Это осуществляется с помощью двух ограничений:

- при добавлении записей в таблицу проверяется уникальность их первичных ключей;
- не допускается изменение значений атрибутов, входящих в первичный ключ.

Требование ссылочной целостности состоит в следующем:

- для каждого значения внешнего ключа, появляющегося в дочернем отношении, в родительском отношении должен найтись кортеж с таким же значением первичного ключа.

Как правило, поддержание ссылочной целостности также возлагается на СУБД.

## **1.2.3 Манипуляционная часть реляционной модели**

Манипуляционная часть реляционной модели описывает два эквивалентных способа манипулирования реляционными данными – реляционную алгебру и реляционное исчисление. Принципиальное различие между реляционной алгеброй и реляционным исчислением заключается в следующем: реляционная алгебра в явном виде предоставляет набор операций, а реляционное исчисление представляет систему обозначений для определения требуемого отношения в терминах данных отношений.

Формулировка запроса в терминах реляционной алгебры носит предписывающий характер, а в терминах реляционного исчисления – описательный характер. Говоря неформально, реляционная алгебра носит процедурный характер (пусть на очень высоком уровне), а реляционное исчисление – непроцедурный характер.

## **1.3 СУБД**

Система управления базами данных (СУБД) - приложение, обеспечивающее создание, хранение, обновление и поиск информации в базах данных.

Основные функции СУБД:

- непосредственное управление данными во внешней памяти;
- управление буферами оперативной памяти;
- управление транзакциями;
- журнализация;
- поддержка языков БД.

## **Вывод**

В данном разделе была приведена формализация задачи, основные принципы реляционных баз данных. В качестве СУБД был выбран PostgreSQL, а в качестве фреймворка Django.



## 2. Конструкторский раздел

В данном разделе будет рассмотрено проектирование ПО, представлены диаграмма вариантов использования, ER диаграмма и диаграмма базы данных. Рассмотрена регистрация и аутентификация пользователей.

### 2.1 Диаграмма вариантов использования

На рисунке 1 представлена Use Case диаграмма с двумя видами акторов.

Администратор: взаимодействует с базой данной, создает, редактирует и удаляет записи, добавляет новых пользователей.

Пользователь: использует приложение для составления списка интересующих его вакансий.



Рисунок 1. Диаграмма вариантов использования

## 2.2 ER диаграмма

На рисунке 2 представлена ER диаграмма приложения.

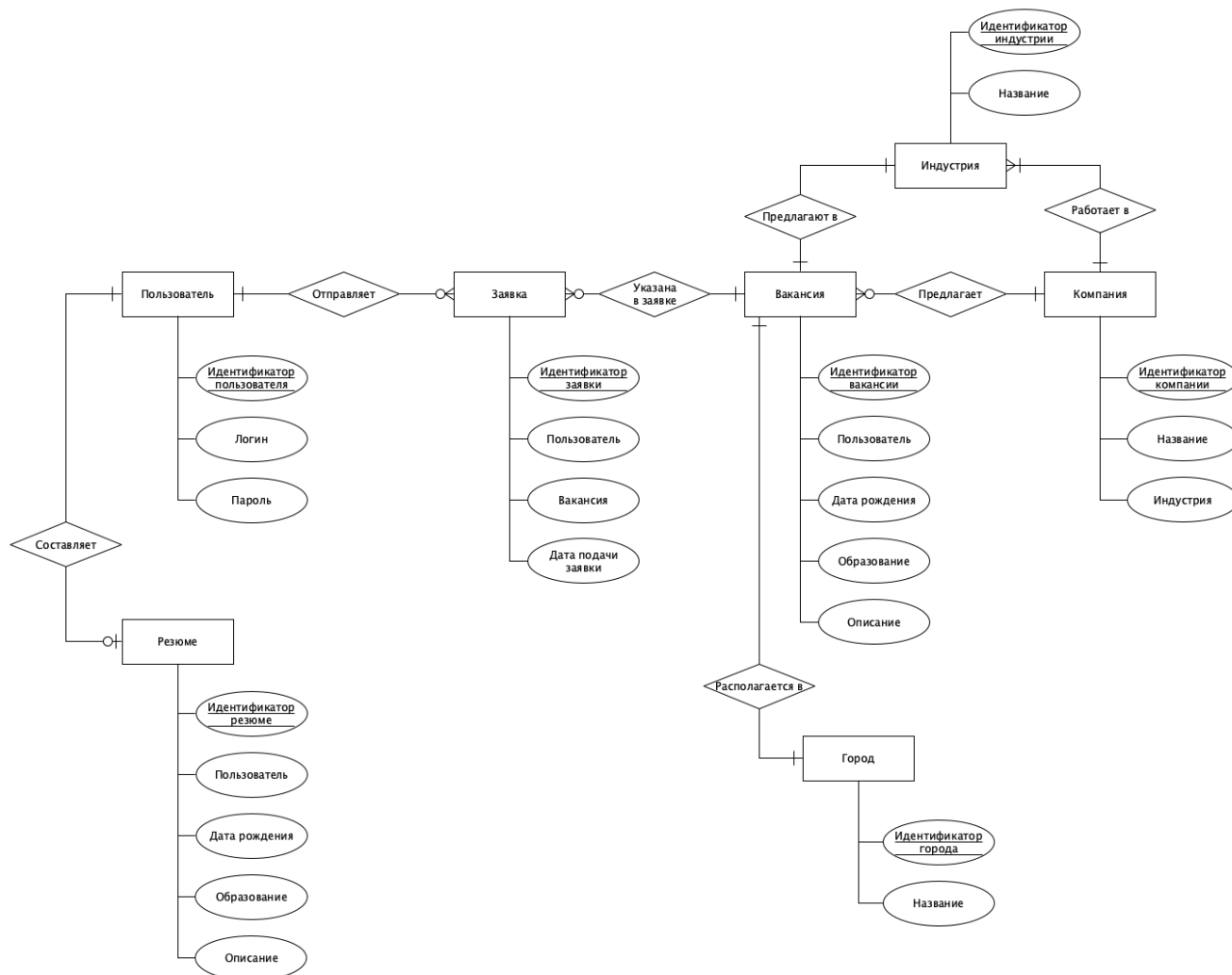


Рисунок 2. ER диаграмма

## 2.3 Проектирование таблиц базы данных

База данных приложения содержит следующие таблицы:

- таблица пользователей User;
- таблица городов City;
- таблица индустрий Industry;
- таблица компаний Company;
- таблица вакансий JobVacancy;

- таблица заявок, поданных пользователями на вакансии, Application;
- таблице резюме пользователей Resume.

### **Таблица User**

Содержит информацию о пользователях сайта.

Таблица включает следующие поля:

- id – целочисленное поле, идентификатор пользователя;
- username – символьное поле, логин пользователя;
- email – символьное поле, адрес электронной почты пользователя;
- password – символьное поле, пароль пользователя.

### **Таблица City**

Содержит информацию о городах.

Таблица включает следующие поля:

- id – целочисленное поле, идентификатор города;
- name – символьное поле, название города.

### **Таблица Industry**

Содержит информацию об индустриях.

Таблица включает следующие поля:

- id – целочисленное поле, идентификатор индустрии;
- name – символьное поле, название индустрии.

### **Таблица Company**

Содержит информацию о компаниях.

Таблица включает следующие поля:

- id – целочисленное поле, идентификатор компании;
- name – символьное поле, название компании;
- industry\_id – целочисленное поле, идентификатор индустрии.

### **Таблица JobVacancy**

Содержит информацию о предлагаемых вакансиях.

Таблица включает следующие поля:

- id – целочисленное поле, идентификатор вакансии;
- company\_id – целочисленное поле, идентификатор компании;
- industry\_id – целочисленное поле, идентификатор индустрии;
- city\_id – целочисленное поле, идентификатор города;
- years\_of\_exp – символьное поле, требуемый опыт работы;
- type – символьное поле, тип занятости на работе.

### **Таблица Application**

Содержит информацию о поданной заявке на вакансию.

Таблица включает следующие поля:

- id – целочисленное поле, идентификатор заявки;
- applied\_on – поле даты, время подачи заявки;
- applicant\_id – целочисленное поле, идентификатор пользователя;
- job\_id – целочисленное поле, идентификатор вакансии.

### **Таблица Resume**

Содержит информацию о резюме пользователя.

Таблица включает следующие поля:

- id – целочисленное поле, идентификатор заявки;
- user\_id – целочисленное поле, идентификатор пользователя;
- birth\_date – поле даты, дата рождения пользователя;
- education – символьное поле, образование пользователя;
- description – целочисленное поле, описание пользователя.

На рисунке 3 представлена диаграмма базы данных.

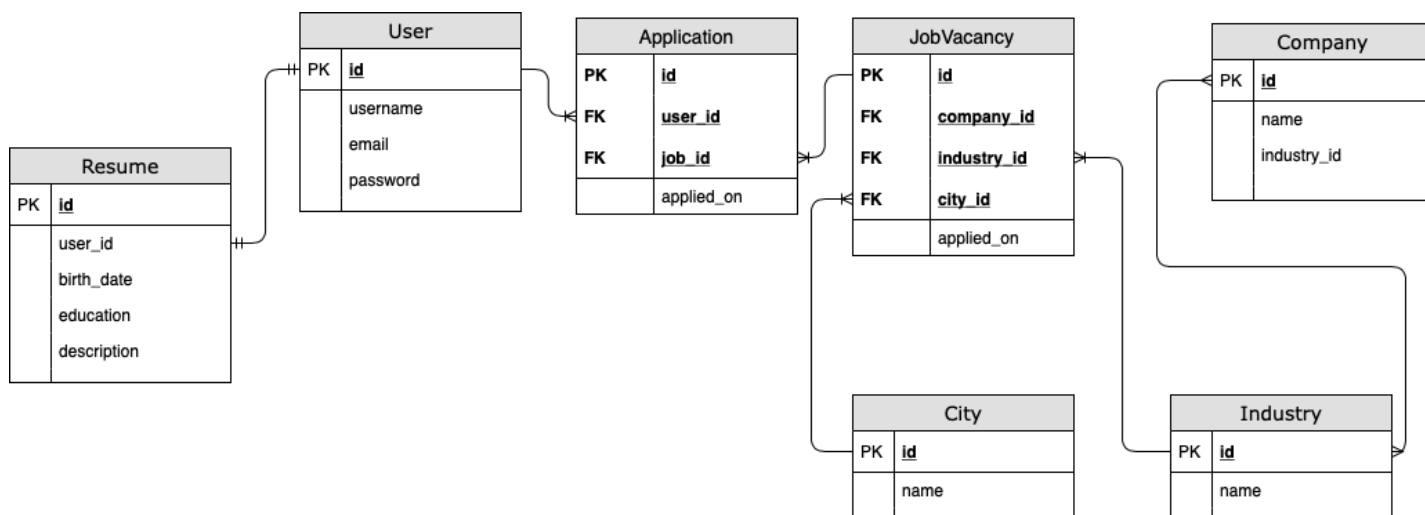


Рисунок 3. Диаграмма базы данных

## 2.4 Паттерн модель-представление-контроллер

Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель (Model), Представление (View) и Контроллер (Controller) – таким образом, что модификация каждого компонента может осуществляться независимо.

Основное преимущество такого подхода заключается в свободе объединения этих компонентов. Следовательно, каждая отдельная часть приложения, созданного с помощью Django, имеет одно назначение и может быть изменена независимо, т.е., без влияния на остальные компоненты.

### 2.4.1 Модели

В Django модели отображают информацию о данных. Они содержат поля и поведение данных, одна модель соответствует одной таблице в базе данных.

В данной работе будет использована технология ORM.

ORM (англ. Object-Relational Mapping, рус. объектно-реляционное

отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

ORM позволяет удобно интегрировать модели в приложения с объектно-ориентированным стилем программирования.

## 2.4.2 Представления

Представление — это компонент системы, нужный для отображения пользователю модели. В Django этим занимаются представления (views) и шаблоны (templates).

В данной работе можно выделить следующие представления:

- страница со списком компаний;
- страница с информацией о выбранной компании;
- страница со списком индустрий;
- страница с информацией о выбранной индустрии;
- страница с информацией о выбранной вакансии;
- страница с формой подачи заявки на вакансию;
- страница поиска вакансии по заданным параметрам;
- страница со списком поданных заявок;
- страница регистрации;
- страница аутентификации.

## 2.4.3 Контроллер

Контроллер определяет представление в зависимости от указаний пользователя. Он перенаправляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

В Django контроллеры обеспечивают обработку HTTP запросов GET и POST. Контроллер есть у каждого представления, так как именно он отправляет пользователю запрашиваемую HTML-страницу.

## **2.5 Регистрация и аутентификация пользователей**

Регистрация пользователя в приложении является добавлением в базу данных (таблица User) записи, содержащей необходимую информацию для аутентификации. Для этого пользователь вводит соответствующие данные в поля регистрационной формы.

Django предоставляет набор базовых инструментов для реализации web-приложения. В этот функционал включена и реализация аутентификации пользователя.

### **Вывод**

В данном разделе была рассмотрена архитектура приложения, представлены диаграммы приложения. Был рассмотрен механизм регистрации и аутентификации пользователей.

### 3. Технологический раздел

В данной части приведены листинги классов для оформления таблиц базы данных, доступ к данным и frontend-разработка, а также рассмотрена интерфейс приложения.

#### 3.1 Выбор технологического стека

В качестве языка программирования был выбран Python. Т.к. он поддерживает разные парадигмы программирование, а также обладает большим количеством фреймворков и библиотек, в том числе для доступа к различным СУБД.

Основными критериями выбора СУБД являлись поддержка реляционной модели данных (т.к. заранее известны типы хранимых данных), наличие ORM и возможность в дальнейшем расширить проект.

Для проекта были рассмотрены две самые популярные СУБД: PostgreSQL и SQLite.

SQLite является компактной встраиваемой БД и допускает единовременное исполнение лишь одной операции записи, в связи с чем не подходит для многопользовательского приложения с большим объемом данных (противоречит критерии дальнейшего развития проекта).

Реляционная СУБД PostgreSQL ориентируется на полное соответствие стандартам и расширяемость, поддерживает одновременную обработку сразу нескольких заданий. Помимо того, PostgreSQL содержит механизм наследование, что позволит в дальнейшем масштабировать проект.

В связи с этим в качестве СУБД в данной работе был выбран PostgreSQL.

СУБД PostgreSQL поддерживается множеством фреймворков, которые содержат в себе необходимые методы обращения к базе данных.

В качестве web-framework был выбран Django, который предоставляет все необходимые инструменты для написания как frontend, так и backend частей для полноценного запуска приложения.



Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других. Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

К основным преимуществам фреймворка Django, которые повлияли на его выбор, стали:

- **быстрота:** Django был разработан, чтобы помочь разработчикам создать приложение настолько быстро, насколько это возможно. Это включает в себя формирование идеи, разработку и выпуск проекта, где Django экономит время и ресурсы на каждом из этих этапов;
- **полная комплектация:** Django работает с десятками дополнительных функций, которые заметно помогают с аутентификацией пользователя, картами сайта, администрированием содержимого, RSS и многим другим. Данные аспекты помогают осуществить каждый этап веб-разработки;
- **безопасность:** работая в Django, вы получаете защиту от ошибок, связанных с безопасностью и ставящих под угрозу проект;
- **масштабируемость:** фреймворк Django наилучшим образом подходит для работы с самыми высокими трафиками. Следовательно, логично, что великое множество загруженных сайтов используют Django для удовлетворения требований, связанных с трафиком.

Для реализации проекта были выбраны следующие технологии:

- язык программирования Python;

- СУБД PostgreSQL;
- framework Django.

Выбранные инструменты совместимы друг с другом и позволяют выполнить все поставленные задачи.

## 3.2 Реализация хранения данных

В листингах 1-6 представлены реализованные модели.

В качестве модели User была взята встроенная модель пользователя Django.

Листинг 1. Модель City

```
class City(models.Model):
    name = models.CharField(max_length = 50)

    def __str__(self):
        return self.name
```

Листинг 2. Модель Industry

```
class Industry(models.Model):
    name = models.CharField(max_length = 50)

    class Meta:
        ordering = ['name']

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('industry-detail', args=[str(self.id)])
```

Листинг 3. Модель Company

```
class Company(models.Model):
    name = models.CharField(max_length = 50)
    industry = models.ManyToManyField(Industry)

    class Meta:
        ordering = ['name']

    def __str__(self):
        return self.name
```

```
def get_absolute_url(self):
    return reverse('company-detail', args=[str(self.id)])
```

Листинг 4. Модель JobVacancy

```
class JobVacancy(models.Model):
    title = models.CharField(max_length = 100)
    company = models.ForeignKey('Company', on_delete=models.SET_NULL, null=True)
    industry = models.ForeignKey('Industry', on_delete=models.SET_NULL, null=True)
    city = models.ForeignKey('City', on_delete=models.SET_NULL, null=True)

    YEARS_OF_EXP = (
        ('entry', 'Entry level'),
        ('1-2', '1-2 years'),
        ('3-5', '3-5 years'),
        ('6-10', '6-10 years'),
        ('above 10', 'Above 10 years')
    )

    years_of_exp = models.CharField('Years of Experience', max_length=20,
    choices=YEARS_OF_EXP, null=True, blank=True)

    TYPES = (
        ('fulltime', 'Full-Time'),
        ('parttime', 'Part-Time')
    )

    type = models.CharField(max_length=10, choices=TYPES)

    objects = JobVacancyManager()

    def __str__(self):
        return self.title

    def save(self, *args, **kwargs):
        self.slug = slugify(self.title)
        super(JobVacancy, self).save(*args, **kwargs)

    def get_absolute_url(self):
        return reverse('job-vacancy-detail', args=[str(self.id)])
```

Листинг 5. Модель Application

```
class Application(models.Model):
    applicant = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    job = models.ForeignKey('JobVacancy', on_delete=models.SET_NULL, null=True)
    applied_on = models.DateTimeField(auto_now_add=True)
```

```
def __str__(self):
    return self.applicant.last_name

def get_absolute_url(self):
    return reverse('application-detail', args=[str(self.id)])
```

Листинг 6. Модель Resume

```
class Resume(models.Model):
    user = models.OneToOneField(User, on_delete=models.SET_NULL, null=True, related_name=
    = 'resume')
    birth_date = models.DateTimeField(auto_now_add=True)
    education = models.CharField(max_length=50)
    description = models.TextField(max_length=1000)

    def __str__(self):
        return self.user.last_name

    def get_absolute_url(self):
        return reverse('resume-detail', args=[str(self.id)])
```

### 3.3 Реализация доступа к данным

Чтобы обеспечить доступ к данным, необходимо создать форму, позволяющую добавлять и изменять записи в таблицах.

Центр данного механизма – класс Form, который описывает структуру объекта, его поведение и представление.

Реализация формы для доступа к данным представлена в листинге 7.

Листинг 7. Форма для регистрации

```
class RegisterForm(forms.Form):
    username = forms.CharField(label="Username",
    max_length=50,min_length=3,error_messages=error_username)
    email = forms.EmailField()
    password = forms.CharField(widget=forms.PasswordInput, label="Password")
    rep_password = forms.CharField(label="Repeat password",
    widget=forms.PasswordInput)
```

## 3.4 Frontend-разработка

Дизайн сайта настроен с помощью технологии Bootstrap.

Bootstrap — это инструмент с открытым исходным кодом для разработки web-приложений с помощью HTML, CSS и JS. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Django предоставляет инструмент шаблонизатора, который дает возможность вносить динамические данные в html с backend. С помощью шаблонизатора есть возможность проверять данные, изменяя элементы страницы в зависимости от результата проверки.

При рендеринге шаблона переменные в двойных фигурных скобках будут заменяться на вычисленные значения.

Например, в шаблоне поиска, представленном в листинге 8, шаблонизатор `{{ form|crispy }}` вставляет на страницу созданную форму поиска, а с использованием шаблонизатора `{% for job in jobs %}` на страницу выводятся все найденные вакансии.

Листинг 8. Шаблон поиска

```
{% extends "base_generic.html" %}
{% load crispy_forms_tags %}
{% block content %}
    <div class="row">

        <div class="col">
            <div class="jobvacancy">
                <div><h1> Search results </h1></div>
                {% for job in jobs %}
                    <li>
                        <a href="{{ job.get_absolute_url }}">{{ job.title }}</a>
                    </li>
                {% endfor %}
            </div>
        </div>
        <div class="col-3">
            <div class="p-3">

                <div><h1> Parameters </h1></div>
                <form action="{% url 'search' %}" method="get" novalidate>
```

```

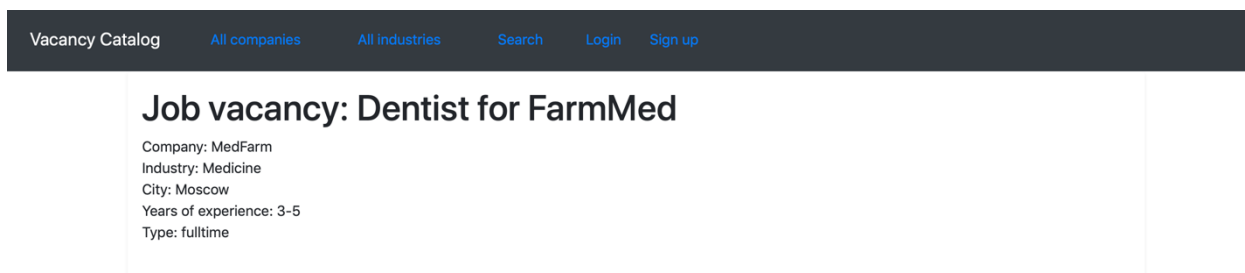
{% csrf_token %}
<table>
  {{ form|crispy }}
</table>
<button type="submit" class="btn btn-primary m-2">Search</button>
</form>
</div>
</div>
</div>
{% endblock %}

```

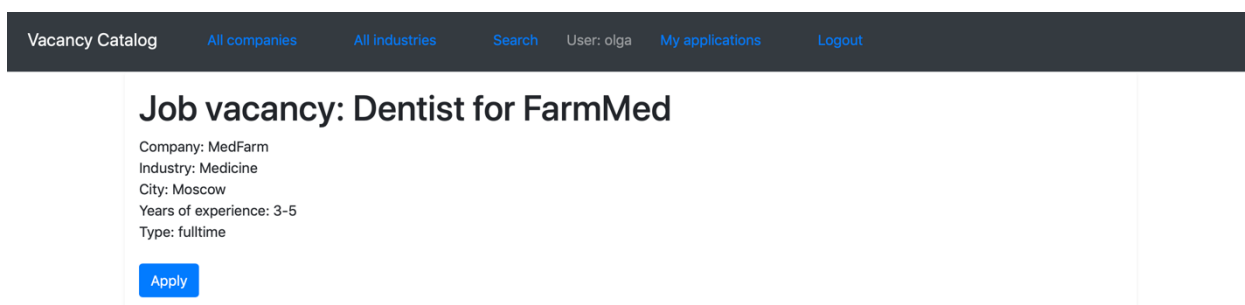
### 3.5 Интерфейс приложения

В зависимости от того, выполнил ли пользователь аутентификацию, интерфейс выглядит по-разному. Зарегистрированному пользователю доступна возможно подать заявка на вакансию, а также доступен список всех вакансий, на которые он откликнулся. Незарегистрированный пользователь не может этого увидеть до тех пор, пока не выполнит вход в систему.

На рисунках 4 и 5 представлен интерфейс зарегистрированного и незарегистрированного пользователей соответственно.

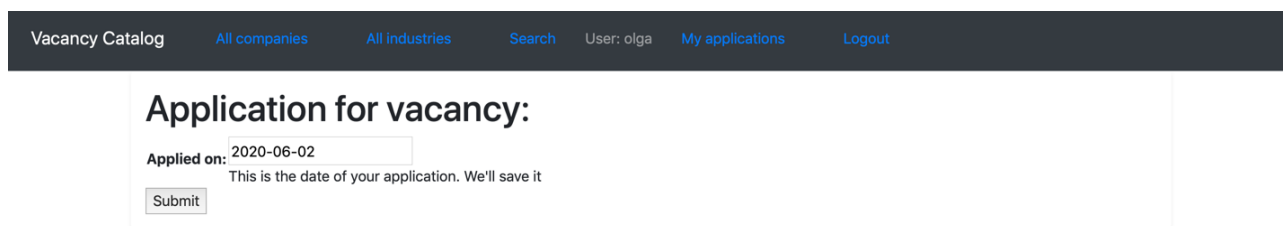


*Рисунок 4. Интерфейс зарегистрированного пользователя*



*Рисунок 5. Интерфейс незарегистрированного пользователя*

Когда пользователь нажимает на кнопку «Apply», приложение запоминает дату подачу заявки (сегодняшнюю дату) и просит еще раз подтвердить отправку, как продемонстрировано на рисунке 6.

The screenshot shows a dark navigation bar at the top with links: 'Vacancy Catalog', 'All companies', 'All industries', 'Search', 'User: olga', 'My applications', and 'Logout'. Below the navigation bar is a white box titled 'Application for vacancy:'. Inside this box, there is a label 'Applied on:' followed by a text input field containing '2020-06-02'. Below the input field is a message: 'This is the date of your application. We'll save it'. At the bottom of the box is a 'Submit' button.

*Рисунок 6. Интерфейс отправки заявки на вакансию*

На рисунках 7 и 8 представлены страницы аутентификации и регистрации пользователей.

Please login to see this page.

Username:

Password:

[Lost password?](#)

*Рисунок 7. Страница аутентификации*

# Registration

Username\*

Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Email\*

Password\*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation\*

Enter the same password as before, for verification.

Save

[Have account already? Login](#)

Рисунок 8. Страница регистрации

На рисунках 9 и 10 представлен интерфейс поиска вакансии по заданным параметрам.

## Search results

- [Dentist for FarmMed](#)
- [Commercial horticulturist](#)
- [Arboriculturist](#)
- [International aid/development worker](#)
- [Retail buyer](#)
- [Manufacturing systems engineer](#)
- [Building control surveyor](#)
- [Facilities manager](#)
- [Operational investment banker](#)
- [Academic librarian](#)
- [Clinical biochemist](#)
- [Animal nutritionist](#)
- [Technical brewer](#)
- [Commercial horticulturist](#)
- [Personnel officer](#)
- [Administrator, charities/voluntary organisations](#)
- [Cytogeneticist](#)
- [Dancer](#)
- [Immigration officer](#)
- [Building services engineer](#)
- [Equities trader](#)
- [Financial trader](#)
- [Medical sales representative](#)
- [Teacher, English as a foreign language](#)
- [Radiographer, diagnostic](#)

## Parameters

Title

Industry

- ☐ Administrative staff
- ☐ Banks, investments
- ☐ Beauty Salons
- ☐ Car business
- ☐ Construction, Real estate
- ☐ Education
- ☐ Home staff
- ☐ Information Technology
- ☐ Insurance
- ☐ Jurisprudence
- ☐ Management
- ☐ Manufacturing
- ☐ Marketing, advertising, PR
- ☐ Mass media
- ☐ Medicine
- ☐ Public service
- ☐ Raw material extraction
- ☐ Sales
- ☐ Security
- ☐ Sports, Fitness

Рисунок 9. Страница поиска до введения параметров



## Search results

- [Dentist for FarmMed](#)

## Parameters

Title

Industry

- ☐ Administrative staff
- ☐ Banks, investments
- ☐ Beauty Salons
- ☐ Car business

*Рисунок 10. Страница со списком найденных результатов*

## Вывод

Были рассмотрены листинги реализованных классов для оформления таблиц базы данных, доступ к данным и frontend-разработка. Был рассмотрен интерфейс приложения и его основные функции.

## **Заключение**

В ходе проделанной работы были проанализированы основные принципы реляционных баз данных и реляционные СУБД.

Спроектирована база данных, состоящая из нескольких сущностей..

С помощью выбранных технологий было реализовано приложение для взаимодействия с базой данных.

## Список литературы

1. Документация Python 3 [Электронный ресурс]. – Режим доступа: URL: <https://docs.python.org/3/> (Дата Обращения - 30.05.2020)
2. Документация к PostgreSQL 12.2 [Электронный ресурс].- Режим доступа: URL: <https://postgrespro.ru/docs/postgresql/12/index.html> (дата обращения: 30.05.2020).
3. Паттерн MVC [Электронный ресурс].- Режим доступа: URL: [https://professorweb.ru/my/WPF/documents\\_WPF/level36/36\\_3.php](https://professorweb.ru/my/WPF/documents_WPF/level36/36_3.php) (дата обращения: 31.05.2020).
4. Документация к Django [Электронный ресурс].- Режим доступа: URL: <https://docs.djangoproject.com/en/3.0/> (дата обращения: 31.05.2020).
5. Документация к Bootstrap [Электронный ресурс].- Режим доступа: URL: <https://getbootstrap.com/docs/4.5/getting-started/introduction/> (дата обращения: 31.05.2020).