Государственное бюджетное образовательное учреждение высшего образования
Московской области
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
Колледж космического машиностроения и технологий

ОТЧЕТ

по производственной практике ПП.01.01

по профессиональному модулю ПМ.01

«Разработка программных модулей программного обеспечения для компьютерных систем»

Специальность 09.02.03 «Программирование в компьютерных системах»

Выполнил студент(ка) 3 курса группы П2-17

Филиппов Л.

_____ (подпись)

Принял преподаватель

Родичкин П. Ф.

_____ (подпись)

_____ (оценка)

Королев 2020

Введение

На 3 курсе обучения в ККМТ, студентом группы П2-17 Филипповым Леонидом была пройдена производственная практика по модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» в ГБОУ ВО МО «Технологический университет» в Управлении качества образования. Студент получил задание изучить структуру одного из подразделений Управления качества образования - «Центр диагностики качества образования и содействия трудоустройству», разработать приложение для управления информацией о выпускниках университета, вывода данной информации в файл excel.

1.Характеристика объекта практики

1.1. Структура организации, характеристики основных видов деятельности

Организационная структура организации представлена на Рис. 6.1 в Приложении 2.

Организационная структура организации представлена на Рис. 6.2 в Приложении 3.

1.2. Описание структуры управления, где проходит практика

Во главе стоит руководитель управления, в его подчинении находятся 3 подразделения:

1. Отдела мониторинга, лицензирования и аккредитации.
2. Центр диагностики качества образования и содействия трудоустройству.
3. Отдел формирования и реализации дистанционных технологий.

Организационная структура отдела представлена на Рис. 1.2.1.

Рис. 1.2.1. Организационная структура управления качества образования.

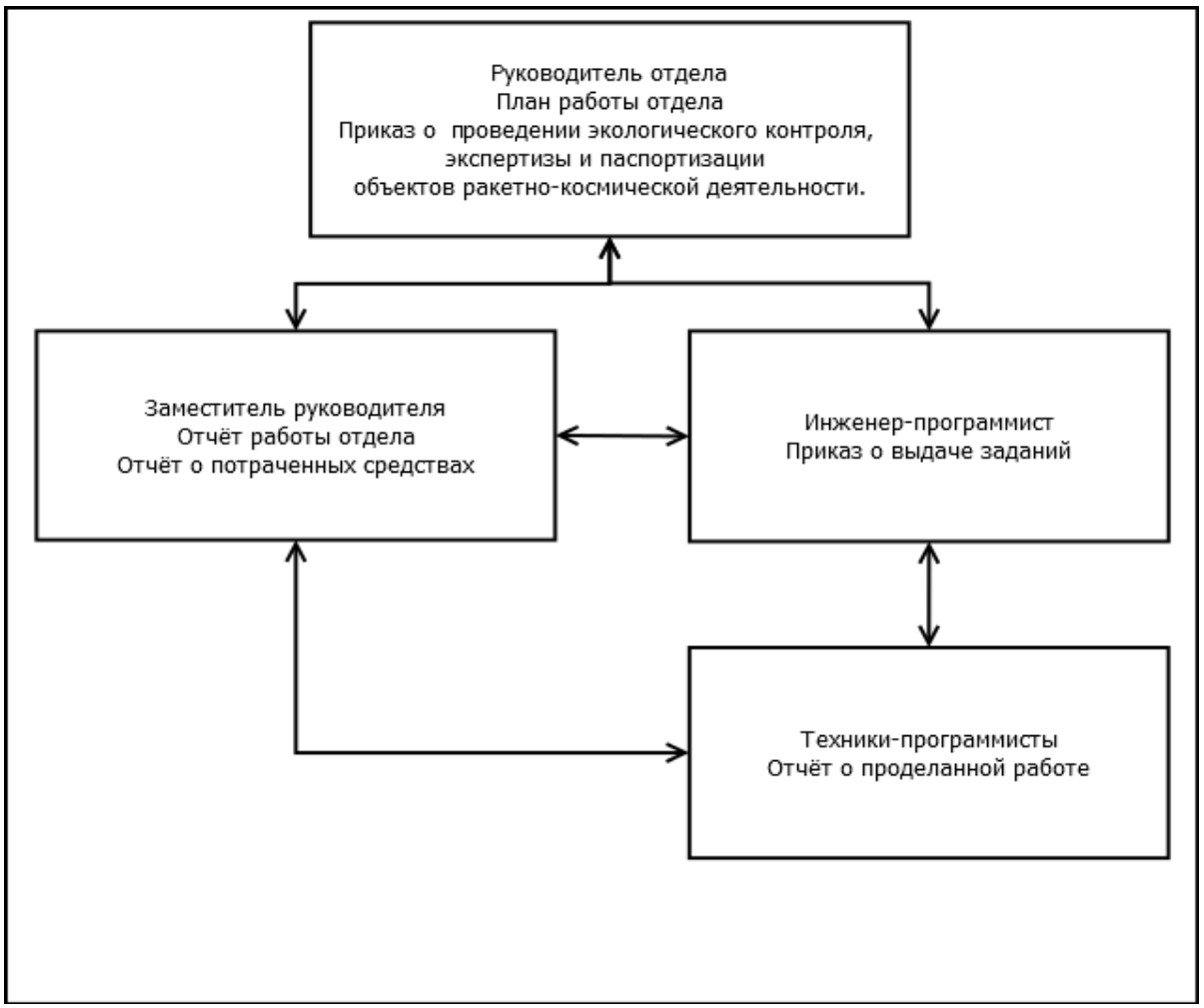

Рис. 1.2.2. Информационная структура отдела

Основные функции управления:

- контроль и управление системой менеджмента качества в университете;
- проведение внутренних аудитов в подразделениях университета;
- обеспечение обслуживания и технической поддержки программного обеспечения для контроля и управления качеством образования;

1.3. Должностные обязанности техника-программиста, инженера-программиста

Ведущий программист:

- На основе анализа математических моделей и алгоритмов решения

экономических и других задач разрабатывает программы, обеспечивающие возможность выполнения алгоритма и, соответственно, поставленной задачи средствами вычислительной техники, проводит их тестирование и отладку.

- Разрабатывает технологию решения задачи по всем этапам обработки информации.
- Осуществляет выбор языка программирования для описания алгоритмов и структур данных.
- Определяет информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля.
- Осуществляет контроль за проведением профилактических работ, устранением неисправностей, возникающих в процессе эксплуатации средств вычислительной техники.
- Определяет объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программ их функциональному назначению.
- Осуществляет запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач.
- Проводит корректировку разработанной программы на основе анализа выходных данных.
- Разрабатывает инструкции по работе с программами, оформляет необходимую техническую документацию.
- Определяет возможности использования готовых программных продуктов.
- Осуществляет сопровождение внедренных программ и программных средств.
- Разрабатывает и внедряет системы автоматической проверки правильности программ, типовые и стандартные программные средства, составляет технологию обработки информации.

- Выполняет работу по унификации и типизации вычислительных процессов.
- Принимает участие в создании каталогов и картотек стандартных программ, в разработке форм документов, подлежащих машинной обработке, в проектировании программ, позволяющих расширить область применения вычислительной техники.
- Выполняет требования системы менеджмента качества, введенные в Университете.

1.4. Документооборот организации (структурного подразделения)

В комплект документов входят следующие виды документов:

- номенклатура дел отдела;
- Положение об Центре;
- политика и Цели в области качества;
- должностные инструкции сотрудников управления (начальников отделов, специалистов по УМР 1 категории, ведущего программиста);
- документы по планированию деятельности отдела;
- перечень нормативной и технической документации (со сведениями об изменениях) и, собственно, документация:

  нормативные правовые акты Российской Федерации, касающиеся сферы среднего профессионального, высшего и дополнительного профессионального образования, регламентирующие образовательную, научную, производственно-хозяйственную деятельность образовательных учреждений;
- Устав Университета;
- приказы и распоряжения ректора Университета;
- инструкции и другие документы Университета;

- перечень видов записей и данных по качеству отдела и, собственно, записи и данные;
- результаты внутренних и внешних аудитов СМК, проведенных в отделе, и выполнения корректирующих и предупреждающих действий;
- результаты проверок состояния помещений и соответствующего оборудования (энергоснабжение, вентиляция и т.д., при необходимости);
- перечень оборудования;
- сведения о ремонтах оборудования, его проверках;
- наличие других локальных актов, определяющих деятельность отдела.

1.5. Состав технических средств обработки данных

Оптический привод DVD-RW LITE-ON IHAS122, внутренний, SATA, черный
SSD накопитель KINGSTON A400 SA400S37/256G 256Гб, 2.5", SATA III
Комплект (клавиатура+мышь) Red Square MK120, USB, проводной, черный
Корпус ATX ACCORD P-26B, Midi-Tower, без БП, черный
Материнская плата ASUS H110M-R/C/SI LGA 1151, mATX
Блок питания FSP Q-DION QD400, 400Вт, 120мм
Модуль памяти CRUCIAL DDR4 - 4Гб, DIMM
Процессор INTEL Pentium Dual-Core G4500, LGA 1151, BOX
Операционная система MICROSOFT Windows 10 Professional OEM
Компьютер INTEL Pentium Dual-Core G4500

1.6. Состав пакетов прикладных программ (ППП) функционального и процедурно-ориентированного типов, виды автоматизированных систем обработки информации и управления

- SQLite
- Django
- Python 3.6

1.7. Характеристика обобщенных технологических процессов сбора, обработки и передачи информации

- Порядок сбора данных в подразделениях:
- Анализ данных, которые запрашиваются;
- Определение ответственных лиц за предоставление данных;
- Рассылка и сбор данных от подразделений;
- Анализ полученных данных и составление отчета.

1.8. Краткая характеристика применяемых методов проектирования подсистем и отдельных задач

Информационные технологий предприятия

- GitHub

Программное обеспечение предприятия

- Linux Ubuntu 18.04
- JetBrains PyCharm Community Edition 2019.2.3.0

Задачи, подлежащие автоматизации

- Создание программных модулей для более быстрого и безошибочного вывода в отчётном виде.
- Упрощение пользовательского интерфейса программы, путём удаления неиспользуемых функций.

2. Разработка отдельных программных модулей и их листинги

2.1 Разработка технического задания

1. Данный проект создан для автоматизации сбора информации о трудоустройстве выпускников.
2. Назначение разработки:
    a. Функциональное назначение программы:

        Функциональным назначением программы является обработка информации, связанная с базами данных.

    b. Эксплуатационное назначение программы:

        Программа может эксплуатироваться в любых условиях. Не требует специального оборудования и дополнительных установок. Пользователями программы могут быть обычные люди, не имеющие специального образования.
3. Требования к программному изделию:
    a. Требования к функциональным характеристикам:
        i. Требования к составу выполняемых функций

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- функция введения данных с клавиатуры;
- функция добавления студентов в БД;
- функция раскрашивания полей студентов;
- функция ввода организаций, вакансий и умений к вакансиям;
- функция данных о выпускнике (ввода университета, специальности и групп);
- функция показа вакансий студентам, если у них совпали умения;
- функция добавления и отображения звонков;
- функция экспорта/импорта в excel;

b. Организация входных и выходных данных

Входные данные программы представлены в виде вводимых с клавиатуры данных.

c. Требования к составу и параметрам технических средств

- браузер

4. Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны быть представлены локализованной версией операционной системы и выше.

2.2 Разработка

Программный код приложения приведен в Приложении 1.

ЗАКЛЮЧЕНИЕ

Перед прохождением производственной практики в Государственном бюджетном образовательном учреждении высшего образования Московской области «Технологический университет» мной были поставлены следующие основные цели:

- приобрести опыт работы по специальности;
- закрепить теоретические знания, полученные во время учебы;
- выполнение требований и действий, предусмотренных программой производственной практики и заданием руководителя;
- закрепить навыки в разработке проектной и технической документации;
- закрепить навыки отладки и тестирования программных модулей.

По окончании практики я достиг своих целей. Для этого я использовал учебную литературу, интернет – источники, документацию средств разработки.

Во время прохождения практики я приобрел опыт работы по специальности. Также был закреплен навык разработки проектной и технической документации и навык отладки и тестирования программных модулей.

По окончании практики был составлен отчет.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. https://docs.djangoproject.com/en/3.0/

Приложение 1

Листинг 1. Пишем файл settings.py для корректной работы сервера

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
MEDIA_DIR = os.path.join(BASE_DIR, 'media')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
# SECRET_KEY = 'w%_hb!y72*eqp6&6)w-rznqz4u@c^de9zu+b&owsiv(ot--e&0'
SECRET_KEY = os.environ.get('DJANGO_SECRET_KEY', 'cg#p$g+j9tax!
#a3cup@1$8obt2_+&k3q+pmu)5%asj6yjpkag')

# SECURITY WARNING: don't run with debug turned on in production!
# DEBUG = True
DEBUG = bool(os.environ.get('DJANGO_DEBUG', True))

ALLOWED_HOSTS = ['kkmt.herokuapp.com', '127.0.0.1']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'students_base',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'base.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'django.template.context_processors.media'
            ],
        },
```

```
    },
]

WSGI_APPLICATION = 'base.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'ru'

TIME_ZONE = 'Europe/Moscow'

USE_I18N = True

USE_L10N = True

USE_TZ = False

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')
MEDIA_URL = '/media/'

# Redirect to home URL after login (Default redirects to /accounts/profile/)
```

```
LOGIN_REDIRECT_URL = '/'

LOGIN_URL = '/accounts/login'

# Heroku: Update database configuration from $DATABASE_URL.
import dj_database_url

db_from_env = dj_database_url.config(conn_max_age=500)
DATABASES['default'].update(db_from_env)
```

Листинг 2. Пишем файл urls.py для обработки запросов по ссылкам

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from django.views.generic import RedirectView


urlpatterns = [
    path('admin/', admin.site.urls),
    path('students_base/', include('students_base.urls')),
    path('', RedirectView.as_view(url='/students_base/', permanent=True)),
]


urlpatterns += [
    path('accounts/', include('django.contrib.auth.urls')),
]


if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Листинг 3. Создаём приложение с помощью команды

```
python manage.py startapp polls
```

Листинг 4. Пишем функции для работы с БД (файл models.py)

```
from django.db import models

IS_BUDGET = (
    ('Бюджет', 'Бюджет'),
    ('Платник', 'Платник')
)
```

```
IS_FREE = (
    ('Свободна', 'Свободна'),
    ('Занята', 'Занята')
)

ON_SPECIALTY = (
    ('По специальности', 'По специальности'),
    ('Не по специальности','Не по специальности'),
)


class College(models.Model):
    """
    Класс для добавления модели College в БД
    """
    title = models.CharField(max_length=100)

    def __str__(self):
        """
        :return: название колледжа
        """
        return self.title


class Specialty(models.Model):
    """
    Класс для добавления модели Specialty в БД
    """
    title = models.CharField(max_length=100)
    college = models.ForeignKey(College, on_delete=models.CASCADE)

    def __str__(self):
        """
        :return: название специальности и название колледжа к которому
        она остностится
        """
        return '{} ({})'.format(self.title, self.college.title)


class Group(models.Model):
    """
    Класс для добавления модели Group в БД
    """
    title = models.CharField(max_length=100)
    specialty = models.ForeignKey(Specialty, on_delete=models.CASCADE)

    def __str__(self):
        """
        :return: название группы, специальности и колледжа
        """
        return '{} ({} - {})'.format(self.title, self.specialty.title,
self.specialty.college.title)


class Job(models.Model):
    """
    Класс для добавления модели Job в БД
    """
    fio = models.CharField(max_length=100)
    release_year = models.CharField(max_length=10)
    employment = models.CharField(max_length=100)
    specialty = models.ForeignKey(Group, on_delete=models.CASCADE)
    practice_one = models.CharField(max_length=100)
```

```
    practice_two = models.CharField(max_length=100)
    budget = models.CharField(max_length=100, choices=IS_BUDGET,
default='Бюджет')
    vacancy_st = models.CharField(max_length=100, null=True, blank=True)
    phone_number = models.CharField(max_length=15)
    email = models.EmailField(max_length=254, help_text='foo@example.com')
    on_speciality = models.CharField(max_length=100, choices=ON_SPECIALTY,
null=True, blank=True)
    expiry_date = models.DateField(null=True, blank=True)
    color = models.CharField(max_length=100, null=True, blank=True)

    def calls_expired_status(self):
        """
        :return: список звонков, у которых стоит статус: 'Истёк'
        """
        return self.calls_set.filter(status='Истёк')

    def calls_not_expired_status(self):
        """
        :return: список звонков, у которых стоит статус: 'В процессе'
        """
        return self.calls_set.filter(status='В процессе')

    def __str__(self):
        """
        :return: имя студента
        """
        return self.fio


class DocumentImg(models.Model):
    """
    Класс для добавления модели DocumentImg в БД
    """
    title = models.CharField(max_length=100)
    document = models.FileField(upload_to='documents/%Y/%m/%d')
    worker = models.ForeignKey(Job, on_delete=models.CASCADE, null=True,
blank=True)
    type = models.CharField(max_length=100)


class List_of_employment(models.Model):
    """
    Класс для добавления модели List_of_employment в БД
    """
    employment = models.CharField(max_length=100)
    color = models.CharField(max_length=100, null=True, blank=True)

    def __str__(self):
        """
        :return: название занятости
        """
        return self.employment


class Organization(models.Model):
    """
    Класс для добавления модели Organization в БД
    """
    title = models.CharField(max_length=100)

    def __str__(self):
        """
```

```
        :return: название организации
        """
        return self.title


class Vacancy(models.Model):
    """
    Класс для добавления модели Vacancy в БД
    """
    title = models.CharField(max_length=100)
    worker = models.OneToOneField(Job, on_delete=models.CASCADE, null=True,
blank=True)
    organization = models.ForeignKey(Organization, on_delete=models.CASCADE,
null=True, blank=True)

    def __str__(self):
        """
        :return: название вакансии
        """
        return self.title


class Skills_Vacancy(models.Model):
    """
    Класс для добавления модели Skills_Vacancy в БД
    """
    vacancy = models.ForeignKey(Vacancy, on_delete=models.CASCADE)
    title = models.CharField(max_length=100)

    def __str__(self):
        """
        :return: название умения для вакансии
        """
        return self.title


class Skills_Student(models.Model):
    """
    Класс для добавления модели Skills_Student в БД
    """
    fio = models.ForeignKey(Job, on_delete=models.CASCADE)
    title = models.CharField(max_length=100)

    def __str__(self):
        """
        :return: название умения студента
        """
        return self.title


class Calls(models.Model):
    """
    Класс для добавления модели Skills_Student в БД
    """
    fio = models.ForeignKey(Job, on_delete=models.CASCADE)
    add_time = models.DateTimeField(auto_now_add=True)
    call_time = models.DateTimeField()
    comment = models.CharField(max_length=150, help_text='max 150 symbols')
    status = models.CharField(max_length=15, null=True, blank=True)
```

Листинг 5. Создаём файл urls.py для управления запросами

```
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.urls import path
from .views import *

urlpatterns = [
    path('', Index.as_view(), name='index'),
    path('vacancy/', VacPage.as_view(), name='vacancy'),
    path('vacancy_skills/', VacSkPage.as_view(), name='vacancy_skills'),
    path('student_skills/', StSkPage.as_view(), name='student_skills'),
    path('add_vacancy/', AddVacancy.as_view(), name='add_vacancy'),
    path('add_org/', AddOrganization.as_view(), name='add_org'),
    path('add_vacancy_org/', AddVacancyOrganization.as_view(),
name='add_vacancy_org'),
    path('add_document/', AddDocument.as_view(), name='add_document'),
    path('student_detail/<int:pk>/', StudentDetail.as_view(),
name='student_detail'),
    path('calls_cancle/<int:pk>/', CancleCalls.as_view(),
name='calls_cancle'),
    path('education/', Education.as_view(), name='education'),
    path('specialty/', Spec.as_view(), name='specialty'),
    path('export/xls/', export_students_xls, name='export_students_xls'),
    path('education/add_group', AddGroup.as_view(), name='add_group'),
    path('import/xls/', import_students_excel, name='import_students_excel'),
    path('settings/', Settings.as_view(), name='settings'),
    path('documentation/', Documentation.as_view(), name='documentation')
]

urlpatterns += staticfiles_urlpatterns()
```

Листинг 6. Создаём файл views.py пишем классы для обработки запросов и взаимодействием с БД

```
import datetime
import xlwt
import xlrd

from django.shortcuts import render, get_object_or_404
from django.http import HttpResponse, Http404
from django.shortcuts import render, redirect
from xlwt.compat import xrange
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator

from .models import *
from django.views.generic import View
from .forms import *


def check_calls(person):
    """
    Функция для проверки статуса звонков
    :param person: студент
    :return: редактирует статусы звонков в БД
    """
    for call in person.calls_set.order_by('-
call_time').filter(call_time__lte=datetime.datetime.now()):
        if call.status != 'Завершён':
            call.status = 'Истёк'
            call.save()
```

```
@method_decorator(login_required, name='dispatch')
class Index(View):
    """
    Класс для отображения главной страницы
    """
    template = 'index.html'

    def get(self, request):
        """
        Функция для обработки get запроса для главной страницы
        :param request: запрос
        :return: рендер главной страницы
        """
        people = Job.objects.order_by("fio")
        list_employments = List_of_employment.objects.order_by("employment")
        for person in people:
            check_calls(person)
        context = {
            'form': PersonForm(),
            'list_employments': list_employments,
            'people': people,
            'imgform': ImageForm,
        }
        return render(request, self.template, context)

    def post(self, request):
        """
        Функция для обработки post запроса для главной страницы
        :param request:
        :return: Если форма пюез ошибок, то redirect на главную иначе рендер
        главной страницы с ошибкой
        """
        bound_form = PersonForm(request.POST)
        is_employment = request.POST.get("employment")
        try:
            el = List_of_employment.objects.get(employment=is_employment)
            color = el.color
        except:
            el = List_of_employment()
            el.employment = is_employment
            el.save()
            color = None
        if bound_form.is_valid():
            bound_form.save()
            people = Job.objects.filter(employment=is_employment)
            for person in people:
                person.color = color
                person.save()
            return redirect('/students_base')
        else:
            print(bound_form.errors)
        context = {
            'form': bound_form
        }
        return render(request, self.template, context)


@method_decorator(login_required, name='dispatch')
class VacPage(View):
    """
```

```
    Класс для отображения страницы с вакансиями
    """
    template = 'vacancy.html'

    def get(self, request):
        """
        Функция для обработки get запроса для страницы с вакансиями
        :param request:
        :return: render страницы с вакансиями
        """
        org = Organization.objects.order_by("title")
        context = {
            'form_org': OrganizationForm(),
            'org_list': org
        }
        return render(request, self.template, context)


class VacSkPage(View):
    """
    Класс для добавления умений для вакансий
    """
    def post(self, request):
        if request.method == 'POST':
            sk_vac = Skills_Vacancy()
            sk_vac.title = request.POST.get("title")
            vac_id = request.POST.get("vacancy")
            vac = Vacancy.objects.get(id=vac_id)
            sk_vac.vacancy = vac
            sk_vac.save()
        return redirect('/students_base/vacancy')


class StSkPage(View):
    """
    Класс для добавления умений для студента
    """
    def post(self, request):
        id_st = request.POST.get("id")
        student = Job.objects.get(id=id_st)
        if request.method == 'POST':
            st_skills = Skills_Student()
            st_skills.fio = student
            st_skills.title = request.POST.get("title")
            st_skills.save()
            return redirect('/students_base/student_detail/' +
str(student.id))
        return HttpResponse(status=404)


class AddVacancy(View):
    """
    Класс для добавления вакансий
    """
    def post(self, request):
        vacancy = request.POST.get("vacancy")
        id_st = request.POST.get("id_st")
        student = Job.objects.get(id=id_st)
        try:
            vac = Vacancy.objects.get(id=vacancy)
            if request.method == 'POST':
                student.vacancy_st = vac.title
```

```
                student.on_speciality = 'По специальности'
                student.save()
                vac.worker = student
                vac.save()
                return redirect('/students_base/student_detail/' +
str(student.id))
        except ValueError:
            return HttpResponse(status=404)


class AddOrganization(View):
    """
    Класс для добавления организаций
    """
    def post(self, request):
        bound_form = OrganizationForm(request.POST)
        if bound_form.is_valid():
            bound_form.save()
            return redirect('/students_base/vacancy')
        return HttpResponse(status=404)


class AddVacancyOrganization(View):
    """
    Класс для добавления вакансий к организациям
    """
    def post(self, request):
        if request.method == 'POST':
            vacancy = Vacancy()
            vacancy.title = request.POST.get("title")
            organization_id = request.POST.get("organization")
            organization = Organization.objects.get(id=organization_id)
            vacancy.organization = organization
            vacancy.save()
            return redirect('/students_base/vacancy')
        return HttpResponse(status=404)


class AddDocument(View):
    """
    Класс для добавления документов
    """
    def post(self, request):
        expansions = ['jpg', 'jpeg', 'png']
        if request.method == 'POST':
            form = ImageForm(request.POST, request.FILES)
            if form.is_valid():
                worker = Job.objects.get(id=request.POST['worker'])
                newimg = DocumentImg(document=request.FILES['docfile'],
worker=worker,
                                     title=request.POST['title'])
                expansion = str(request.FILES['docfile']).split(".")[1]
                newimg.type = 'Картинка' if expansion in expansions else
'Документ'
                newimg.save()
                return redirect('/students_base/')
            else:
                return HttpResponse(status=404)


@method_decorator(login_required, name='dispatch')
```

```
class StudentDetail(View):
    """
    Класс для отображения страницы студента
    """
    template = 'student_detail.html'

    def check_vacancy(self, person):
        """
        Функция для поиска вакансий для студента исходя из умений
        :param person: студент
        :return: вакансии
        """
        vacancy_p = []
        sch = 0
        vacancy = Vacancy.objects.order_by("title")
        people = Job.objects.order_by("fio")
        for vac in vacancy:
            if not vac.worker:
                for key in person.skills_student_set.all():
                    for key_v in vac.skills_vacancy_set.all():
                        if str(key) == str(key_v):
                            sch += 1
                if sch != 0:
                    vacancy_p.append(vac)
        return vacancy_p

    def get(self, request, pk):
        """
        Функция для обработки get запроса для страницы студента
        :param request: запрос
        :param pk: id студента
        :return: render страницы студента
        """
        person = get_object_or_404(Job, id=pk)
        check_calls(person)
        context = {
            'person': person,
            'call_form': CallsForm(),
            'vacancy': self.check_vacancy(person)
        }
        return render(request, self.template, context)

    def post(self, request, pk):
        """
        Функция для обработки post запроса для страницы студента
        :param request: запрос
        :param pk: id студента
        :return: redirect на страницу студента
        """
        person = get_object_or_404(Job, id=pk)
        context = {
            'person': person,
            'call_form': CallsForm()
        }
        if request.method == 'POST':
            time = request.POST.get("call_time").split("T")
            call = Calls()
            call.call_time = ("{} {}").format(time[0], time[1])
            call.comment = request.POST.get("comment")
            call.fio = person
            call.status = 'В процессе'
            call.save()
```

```
        return redirect('/students_base/student_detail/' + str(person.id))


class CancleCalls(View):
    """
    Класс для завершения звонка
    """
    def post(self, request, pk):
        if request.method == 'POST':
            call = get_object_or_404(Calls, id=pk)
            call.status = request.POST.get("status")
            call.save()
            return redirect('/students_base/student_detail/' +
str(call.fio.id))


@method_decorator(login_required, name='dispatch')
class Education(View):
    """
    Класс для отображения страницы образование
    """
    template = 'education.html'

    def get(self, request):
        """
        Функция для обработки post запроса для страницы образование
        :param request: запрос
        :return: render страницы образование
        """
        colleges = College.objects.all()
        context = {
            'form': CollegeForm(),
            'colleges': colleges,
            'g_form': GroupForm(),
        }
        return render(request, self.template, context)

    def post(self, request):
        """
        Функция для обработки post запроса для страницы образование
        :param request: запрос
        :return: redirect на страницу образование, если форма без ошибок
иначе
        страницу 404
        """
        bound_form = CollegeForm(request.POST)
        if bound_form.is_valid():
            bound_form.save()
            return redirect('/students_base/education')
        return HttpResponse(status=404)


class Spec(View):
    """
    Класс для добавления специальности
    """
    def post(self, request):
        if request.method == 'POST':
            college = College.objects.get(id=request.POST.get("college"))
            spec = Specialty()
            spec.title = request.POST.get("title")
```

```
            spec.college = college
            spec.save()
            return redirect('/students_base/education')
        return HttpResponse(status=404)


def export_students_xls(request):
    """
    Функция для экспорта студентов в excel
    :param request: запрос
    :return: запрос
    """
    if request.method == 'POST':
        response = HttpResponse(content_type='application/ms-excel')
        response['Content-Disposition'] = 'attachment;
filename="students.xls"'

        wb = xlwt.Workbook(encoding='utf-8')
        ws = wb.add_sheet('Студенты')

        row_num = 0

        font_style = xlwt.XFStyle()
        font_style.font.bold = True

        columns = []
        print(request.POST.values())
        fields_dict = request.POST.dict()
        del fields_dict["csrfmiddlewaretoken"]

        for field in fields_dict.values():
            columns.append(field)

        # columns = ['ФИО', 'e-mail', 'телефон', 'форма финансирования',
        #            'трудоустроенность', 'вид занятости']

        for col_num in xrange(len(columns)):
            ws.write(row_num, col_num, columns[col_num], font_style)

        # Sheet body, remaining rows
        font_style = xlwt.XFStyle()

        rows = []
        for student in Job.objects.all():
            student_info = []
            if request.POST.get("fio_check") is not None:
                student_info.append(student.fio)
            if request.POST.get("release_year_check") is not None:
                student_info.append(student.release_year)
            if request.POST.get("employment_check") is not None:
                if student.expiry_date is not None:
                    student_info.append(student.employment +
str(student.expiry_date))
                else:
                    student_info.append(student.employment)
            if request.POST.get("specialty_check") is not None:
                student_info.append(student.specialty.title)
            if request.POST.get("practice_one_check") is not None:
                student_info.append(student.practice_one)
            if request.POST.get("practice_two_check") is not None:
                student_info.append(student.practice_two)
```

```
            if request.POST.get("vacancy_st_check") is not None:
                if student.vacancy_st is not None:
                    student_info.append(student.vacancy_st)
                else:
                    student_info.append("Безработный")
            if request.POST.get("on_speciality_check") is not None:
                student_info.append(student.on_speciality)
            rows.append(student_info)

        # rows = Job.objects.all().values_list('fio', 'email',
'phone_number',
        #                                      'budget', 'vacancy_st',
'on_speciality')
        for row in rows:
            row_num += 1
            for col_num in xrange(len(row)):
                ws.write(row_num, col_num, row[col_num], font_style)

        wb.save(response)
        return response


def import_students_excel(request):
    """
    Функция для импорта студентов в excel
    :param request: запрос
    :return: файл excel со студентами
    """
    if request.method == 'POST':
        filename = request.FILES['filename']
        file_import = xlrd.open_workbook(file_contents=filename.read())
        sheet = file_import.sheet_by_index(0)
        for rownum in range(sheet.nrows):
            student = Job()
            student.fio = sheet.row_values(rownum)[0]
            student.release_year = datetime.datetime(
                *xlrd.xldate_as_tuple(sheet.row_values(rownum)[1],
file_import.datemode)).strftime('%d.%m.%Y')
            student.budget = sheet.row_values(rownum)[2]
            student.practice_one = sheet.row_values(rownum)[3]
            student.practice_two = sheet.row_values(rownum)[4]
            student.phone_number = sheet.row_values(rownum)[5]
            student.email = sheet.row_values(rownum)[6]
            student.employment = sheet.row_values(rownum)[7].split("|")[0]
            try:
                el =
List_of_employment.objects.get(employment=sheet.row_values(rownum)
[7].split("|")[0])
                student.color = el.color
            except:
                el = List_of_employment()
                el.employment = sheet.row_values(rownum)[7].split("|")[0]
                el.save()
            if len(sheet.row_values(rownum)[7].split("|")) > 1:
                date = sheet.row_values(rownum)[7].split("|")[1]
                student.expiry_date = datetime.datetime.strptime(date, "%d.
%m.%Y")
            coll = College.objects.get(title=sheet.row_values(rownum)[10])
            spec = coll.specialty_set.get(title=sheet.row_values(rownum)[9])
            gr = spec.group_set.get(title=sheet.row_values(rownum)[8])
            student.specialty = gr
            if len(sheet.row_values(rownum)) == 13:
```

```
                if sheet.row_values(rownum)[11]:
                    student.vacancy_st = sheet.row_values(rownum)[11]
                    student.on_speciality = sheet.row_values(rownum)[12]
            student.save()
        return redirect('/students_base/')


class AddGroup(View):
    """
    Класс для добавления группы
    """
    def post(self, request):
        bound_form = GroupForm(request.POST)
        if bound_form.is_valid():
            bound_form.save()
            return redirect('/students_base/education')
        return HttpResponse(status=404)


@method_decorator(login_required, name='dispatch')
class Settings(View):
    """
    Класс для отображения страницы с настройками
    """
    template = 'settings.html'

    def get(self, request):
        """
        Функция для обработки get запроса для страницы с настройками
        :param request: запрос
        :return: render страницы с настройками
        """
        employments = List_of_employment.objects.all()
        context = {
            'employments': employments,
        }
        return render(request, self.template, context)

    def post(self, request):
        """
        Функция для обработки post запроса для страницы с настройками
        :param request: запрос
        :return: redirect на страницу с настройками
        """
        if request.method == 'POST':
            employment = get_object_or_404(List_of_employment,
id=request.POST.get("employment"))
            employment.color = request.POST.get("color")
            employment.save()
            people = Job.objects.filter(employment=employment)
            for person in people:
                person.color = employment.color
                person.save()
            return redirect('/students_base/settings/')


@method_decorator(login_required, name='dispatch')
class Documentation(View):
    """
    Класс для отображения страницы с документацией
    """
```

```
    template = "documentation.html"

    def get(self, request):
        """
        Функция для обработки get запроса для страницы с документацией
        :param request: запрос
        :return: render страницы с документацией
        """
        context = {
            'title': 'Документация',
        }
        return render(request, self.template, context)
```

Листинг 7. Создаём файл forms.py и пишем туда классы для отображения форм на сайте

```
from django import forms

from .models import *


class PersonForm(forms.ModelForm):
    """
    Класс формы для добавления студентов
    """
    class Meta:
        """
        Класс для определения полей модели Job
        """
        model = Job
        fields = ['fio', 'release_year', 'budget', 'specialty', 'vacancy_st',
'on_speciality',
                  'practice_one', 'practice_two', 'phone_number', 'email',
                  'expiry_date', 'employment']
        labels = {
            'fio': '*ФИО',
            'release_year': '*Год выпуска',
            'employment': '*Занятость',
            'practice_one': '*Место прохождения первой практики',
            'practice_two': '*Место прохождения второй практики',
            'budget': '*Бюджет / Платник',
            'phone_number': '*Номер телефона',
            'email': '*Электронная почта',
            'specialty': '*Специальность',
            'vacancy_st': 'Вакансия',
            'on_speciality': 'Трудоустроенность',
            'expiry_date': 'Если в армии / декрет, то до какого числа',
        }
        widgets = {
            'fio': forms.TextInput(attrs={'class': 'form-control'}),
            'release_year': forms.TextInput(attrs={'class': 'form-control',
'id': 'date_of_birth'}),
            'speciality': forms.TextInput(attrs={'class': 'form-control'}),
            'employment': forms.TextInput(attrs={'class': 'form-control',
'id': 'employment'}),
            'practice_one': forms.TextInput(attrs={'class': 'form-control'}),
            'practice_two': forms.TextInput(attrs={'class': 'form-control'}),
            'budget': forms.RadioSelect(),
            'phone_number': forms.TextInput(attrs={'class': 'form-control',
'id': 'phone_number'}),
            'email': forms.EmailInput(attrs={'class': 'form-control', 'id':
'email'}),
            'specialty': forms.Select(attrs={'class': 'form-control'}),
```

```
            'vacancy_st': forms.TextInput(attrs={'class': 'form-control'}),
            'on_speciality': forms.Select(attrs={'class': 'form-control'}),
            'expiry_date': forms.DateInput(attrs={'class': 'form-control',
'type': 'date'}),
        }


class OrganizationForm(forms.ModelForm):
    """
    Класс формы для добавления организации
    """
    class Meta:
        """
        Класс для определения полей модели Organization
        """
        model = Organization
        fields = '__all__'
        labels = {
            'title': 'Название организации'
        }
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-control'})
        }


class ImageForm(forms.Form):
    """
    Класс формы для добавления файла/картинки документов студента
    """
    docfile = forms.FileField(
        label='Добавить документ',
        help_text='max. 42 megabytes'
    )


class CallsForm(forms.ModelForm):
    """
    Класс формы для добавления звонков
    """
    class Meta:
        """
        Класс для определения полей модели Calls
        """
        model = Calls
        fields = '__all__'
        widgets = {
            'call_time': forms.TextInput(attrs={'class': 'form-control',
'type': 'datetime-local'})
        }


class CollegeForm(forms.ModelForm):
    """
    Класс формы для добавления колледжей
    """
    class Meta:
        """
        Класс для определения полей модели College
        """
        model = College
        fields = '__all__'
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-control'})
```

```
        }
        labels = {
            'title': 'Название университета/спо'
        }


class GroupForm(forms.ModelForm):
    """
    Класс формы для добавления группы
    """
    class Meta:
        """
        Класс для определения полей модели Group
        """
        model = Group
        fields = '__all__'
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-control'}),
            'specialty': forms.Select(attrs={'class': 'form-control'}),
        }
        labels = {
            'title': 'Название группы',
            'specialty': 'Название специальности'
        }
```

Листинг 8. Создаём файл admin.py и пишем код для отображения админской панели

```
from django.contrib import admin
from .models import *

# Регистрация моделей в админ панели
admin.site.register(Job)  # модель студентов
admin.site.register(Organization)  # модель организаций
admin.site.register(Vacancy)  # модель вакансий
admin.site.register(College)  # модель учебных заведений
admin.site.register(Specialty)  # модель специальностей
admin.site.register(Group)  # модель групп
```

Листинг 9. Создаём папку templates и добавляем туда html страницы

Листинг 10. Создаём файл base_generic.html и пишем код который повторяется во всех страницах туда, а потом наследуемся от этого шаблона

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no">
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.cs
s"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q
9Ifjh" crossorigin="anonymous">
<script src="http://code.jquery.com/jquery-1.8.3.js"></script>
{% block title %}<title>Local Library</title>{% endblock %}
</head>
<body>
<nav class="navbar navbar-expand-lg sticky-top navbar-light bg-light">
  <a class="navbar-brand" href="{% url 'index' %}">Главная</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-
expanded="false" aria-label="Toggle navigation">
```

```
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
    <div class="navbar-nav">
      <a class="nav-item nav-link" href="{% url 'vacancy' %}">Работа</a>
      <a class="nav-item nav-link" href="{% url 'education' %}">Образование</
a>
    </div>
  </div>
  <div class="form-inline">
    <a class="nav-item nav-link text-muted" href="" data-toggle="modal" data-
target="#ExcelModel">Excel</a>
    <a class="nav-item nav-link text-muted" href="{% url 'settings'
%}">Настройки</a>
    <a class="nav-item nav-link text-muted" href="{% url 'documentation'
%}">Документация</a>
    {% if user.is_authenticated %}
     <a class="nav-item nav-link text-muted" href="{% url 'logout'%}?
next={{request.path}}">Выйти</a>
   {% else %}
     <a class="nav-item nav-link text-muted" href="{% url 'login'%}?
next={{request.path}}">Войти</a>
   {% endif %}
  </div>
</nav>
{% block content %} {% endblock %}
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery.maskedinput/1.4.1/
jquery.maskedinput.min.js"></script>
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
</body>
</html>
```

Листинг 11. Создаём файл documentation.html пишем код для отображения страницы с документацией

```
{% extends "base_generic.html" %}

{% block title %}
<title>{{ title }}</title>
{% endblock %}

{% block content %}
{% load static %}

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
bg-light">
```

```
  <div class="col-md-5 p-lg-5 mx-auto my-5">
    <h1 class="display-8 font-weight-normal">Добавление студентов в базу</h1>
    <p class="lead font-weight-normal">Их можно добавить двумя способами,
через форму на главной странице или через файл excel</p>
  </div>
</div>

<div class="container">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center text-
white overflow-hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление через форму на главной</h2>
      <p class="lead">Поля без звёздочек не обязательны</p>
    </div>
    <img src="{% static 'images/form.png' %}" alt="Добавление через форму на
главной" class="img-thumbnail">
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-white overflow-
hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление списком через excel файл</h2>
      <p class="lead">Примечание: при указании группы, специальности и
университета - сначала они <span class="text-danger">обязательно</span>
должны быть добавлены во вкладке образование</p>
      <p class="lead">Пример файла excel для загрузки (его также можно
скачать во вкладке excel):</p>
    </div>
    <img src="{% static 'images/example_import_excel.png' %}" alt="Добавление
списком через excel файл" class="img-thumbnail">
    <p class="lead">1 колонка - ФИО, 2 - год выпуска, 3 - Бюджет / Платник, 4
- первое место прохождения практики,
    5 - второе место прохождения практики, 6 - номер телефона, 7 - email, 8 -
группа (<span class="text-danger"> см. примечание</span>),
    9 - специальность (<span class="text-danger"> см. примечание</span>), 10
- университет (<span class="text-danger"> см. примечание</span>),
    11 - вакансия (может быть не указана, тогда на главной будет написано
безработный), 12 - по специальности или нет (тоже можно не указывать, если не
указали вакансию)</p>
  </div>
</div>

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
bg-light">
  <div class="col-md-5 p-lg-5 mx-auto my-5">
    <h1 class="display-8 font-weight-normal">Добавление организаций, вакансий
и умений к вакансиям в базу</h1>
    <p class="lead font-weight-normal">Они добавляются во вкладке работа</p>
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-white overflow-
hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление организации</h2>
      <p class="lead">Вводите название организации в поле и нажимаете
добавить / enter</p>
    </div>
```

```
    <img src="{% static 'images/added_organization.png' %}" alt="Добавление
организации" class="img-thumbnail">
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center text-
white overflow-hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление вакансии</h2>
      <p class="lead">Организация появится в списке ниже и под ней есть
кнопка добавить, нажимаете её
       и вылезает окно с полем куда надо вписать название вакансии и нажать
enter</p>
    </div>
    <img src="{% static 'images/added_vacancy.png' %}" alt="Добавление
вакансии" class="img-thumbnail">
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center text-
white overflow-hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление умений для вакансии</h2>
      <p class="lead">После добавления вакансии её карточка появится снизу.
      В ней будет поле, в которое надо вписать умение и нажать enter</p>
    </div>
    <img src="{% static 'images/added_skills_for_vacancy.png' %}"
alt="Добавление умений для вакансии" class="img-thumbnail">
  </div>
</div>

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
bg-light">
  <div class="col-md-5 p-lg-5 mx-auto my-5">
    <h1 class="display-8 font-weight-normal">Добавление университетов,
специальностей и групп в базу</h1>
    <p class="lead font-weight-normal">Они добавляются через форму во вкладке
образование (после добавления при добавлении студента можно будет выбрать
группу из списка)</p>
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-white overflow-
hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление университетов</h2>
      <p class="lead">В форме образование в поле "Добавить университет / спо"
вписать название и нажать добавить/enter</p>
    </div>
    <img src="{% static 'images/added_university.png' %}" alt="Добавление
университетов" class="img-thumbnail">
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-white overflow-
hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление группы к специальности</h2>
```

```
      <p class="lead">В форме добавить группу написать название группы и из
выпадающего списка выбрать группу,
        в скобках будет университет/спо относящийся к этой группе</p>
    </div>
    <img src="{% static 'images/added_group.png' %}" alt="Добавление группы к
специальности" class="img-thumbnail">
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center text-
white overflow-hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление специальности</h2>
      <p class="lead">Снизу от форм будет список университетов, в них список
специальностей, групп и студентов относящихся к этой группе (при нажатии на
ФИО вы перейдёте на страницу студента)</p>
    </div>
    <img src="{% static 'images/added_specialty.png' %}" alt="Добавление
специальности" class="img-thumbnail">
  </div>
</div>

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
bg-light">
  <div class="col-md-5 p-lg-5 mx-auto my-5">
    <h1 class="display-8 font-weight-normal">Настройка цветов полей</h1>
    <p class="lead font-weight-normal">Их можно добавить/исправить во вкладке
настройки</p>
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-white overflow-
hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление цвета полей</h2>
      <p class="lead font-weight-normal">В поле выбираете из списка занятось,
в другом поле выбераете цвет и нажимаете добавить
        (<span class="text-danger">при добавлении новых занятостей они
изначально будут булые в списке цветов</span>)</p>
    </div>
    <img src="{% static 'images/added_color.png' %}" alt="Добавление цвета
полей" class="img-thumbnail">
  </div>
</div>

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
bg-light">
  <div class="col-md-5 p-lg-5 mx-auto my-5">
    <h1 class="display-8 font-weight-normal">Добавление документов к
студентам</h1>
    <p class="lead font-weight-normal">Их можно добавить нажав кнопку
документы в таблице</p>
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center text-
white overflow-hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление документа</h2>
```

```
      <p class="lead font-weight-normal">В поле навание документа пишите
название, в другом поле прикрепляете файл с компьютера
      (<span class="text-danger">файл может быть любого формата, если это
картинка (.jpg, .jpeg, .png), то она там отобразится, если нет,
        то отобразится название, нажав по которому скачается файл который вы
загружали</span>)</p>
    </div>
    <img src="{% static 'images/added_document.png' %}" alt="Добавление
документа" class="img-thumbnail">
    <img src="{% static 'images/list_document.png' %}" alt="Добавление
документа" class="img-thumbnail mt-5">
  </div>
</div>

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
bg-light">
  <div class="col-md-5 p-lg-5 mx-auto my-5">
    <h1 class="display-8 font-weight-normal">Страница студента, добавление
звонков и умений студента</h1>
    <p class="lead font-weight-normal">На страницу можно перейти нажав на
кнопку перейти в таблице</p>
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center text-
white overflow-hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление умений</h2>
      <p class="lead font-weight-normal">Перейдя на страницу студента вы
увидите всю информацию про него,
      пролистав ниже вы увидите его вакансию (если при добавлении студента вы
её указали,
        если нет, то в таблице у него будет написано "безработный", если его
умения
        совпадают с умениями вакансии (со страници с вакансиями), то ему
предложится эта вакансия,
        если ему назначить эту вакансию, она будет не доступна для других и
        на странице с вакансиями будет видно кому она принадлежит) или
добавление умений.
      в поле надо написать название умения и нажать добавить/enter</p>
    </div>
    <img src="{% static 'images/added_skills_for_student.png' %}"
alt="Добавление умений" class="img-thumbnail">
    <img src="{% static 'images/vacancy_name.png' %}" alt="Добавление умений"
class="img-thumbnail mt-5">
  </div>
</div>

<div class="container mt-2">
  <div class="bg-dark mr-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center text-
white overflow-hidden">
    <div class="my-3 py-3">
      <h2 class="display-5">Добавление звонков</h2>
      <p class="lead font-weight-normal">В самом низу на странице студента
будет форма добавления звонков,
      где можно выбрать дату звонка и описание. Если кнопка подробнее стала
красной, значит один из
      звонков истёк (т.е в это время нужно позвонить), этот звонок, на
странице студента будет синим,
      если нажать по кнопке завершить, звонок уберётся; заонок будет серым
если он ещё не истёк.</p>
```

```
    </div>
    <img src="{% static 'images/calls.png' %}" alt="Добавление звонков"
class="img-thumbnail">
  </div>
</div>
{% endblock %}
```

Листинг 12. Создаём файл index.html и пишем туда код для отображения главной страницы

```
{% extends "base_generic.html" %}

{% block title %}
<title>База студентов</title>
{% endblock %}

{% block content %}
{% load static %}

<div class="container mt-5">
    <form action="{% url 'index' %}" method="POST">
            {% csrf_token %}
        {% for field in form %}
            <div class="form-group row">
                <div class="col-sm-10">
                        {{ field.label }}
                        {{ field }}
                </div>
                </div>
        {% endfor %}
        <select id="select_employment" class="form-control w-50">
          <option value="Другое">Выбор занятости</option>
          {% for el in list_employments %}
            <option value="{{el}}">{{el}}</option>
          {% endfor %}
        </select>
        <input type="submit" value="Добавить" class="btn btn-primary mt-2">
    </form>
</div>

        <div class="container mt-5">
        <input class="form-control" id="myInput" type="text"
placeholder="Search..">
        </div>
        <table class="table mt-5" id="to_print">
          <thead>
            <tr>
              <th scope="col">ФИО</th>
              <th scope="col">Год выпуска</th>
              <th scope="col">Шифр / Наименование специальности</th>
              <th scope="col">Бюджет / Внебюджет</th>
              <th scope="col">Место прохождения первой практики</th>
              <th scope="col">Место прохождения второй практики</th>
              <th scope="col">Занятость</th>
              <th scope="col">Вакансия</th>
            </tr>
          </thead>
          <tbody id="myTable">
            {% for person in people %}
            <tr>
```

```
            <td style="color: {{ person.color }};">{{ person.fio }}</td>
            <td style="color:
{{ person.color }};">{{ person.release_year }}</td>
            <td style="color: {{ person.color }};">{{ person.specialty }}</
td>
            <td style="color: {{ person.color }};">{{ person.budget }}</td>
            <td style="color:
{{ person.color }};">{{ person.practice_one }}</td>
            <td style="color:
{{ person.color }};">{{ person.practice_two }}</td>
            <td style="color: {{ person.color }};">{% if person.employment
== 'Декрет' or person.employment == 'Уход в армию'%}{{ person.employment }}
(до {{ person.expiry_date }}){% else %}{{ person.employment }}{% endif
%}</td>
            {% if not person.vacancy_st == NULL %}
            <td class="{% if person.on_speciality == 'По специальности' or
person.vacancy %}text-primary{% else %}text-dark{% endif %}">{% if not
person.vacancy %} {{ person.vacancy_st }} {% else %} {{ person.vacancy }} {%
endif %}</td>
            {% else %}
            <td class = "text-warning">Безработный</td>
            {% endif %}
            <td>
                <button type="button" class="btn {% if
person.documentimg_set.all.count > 0 %}btn-success{% else %}btn-primary{%
endif %}" data-toggle="modal" data-target="#Modal{{ person.id }}">
                  Документы
                </button>

                <div class="modal fade" id="Modal{{ person.id }}"
tabindex="-1" role="dialog" aria-labelledby="ModalLabel" aria-hidden="true">
                  <div class="modal-dialog" role="document">
                    <div class="modal-content">
                      <div class="modal-header">
                        <h5 class="modal-title"
id="ModalLabel">Подтверждающие документы</h5>
                        <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                          <span aria-hidden="true">&times;</span>
                        </button>
                      </div>
                      <div class="modal-body">
                          {% if person.documentimg_set.all %}
                            {% for img in person.documentimg_set.all %}
                                {% if img.type == 'Картинка' %}
                                    <h3>{{img.title}}:</h3>
                                    <img src="{{ MEDIA_URL }}
{{ img.document }}" alt="{{img.title}}" class="img-thumbnail">
                                    <hr>
                                {% else %}
                                    <a href="{{ MEDIA_URL }}
{{ img.document }}">{{img.title}}</a>
                                    <hr>
                                {% endif %}
                            {% endfor %}
                          {% else %}
                            <p>Документов пока нету :)</p>
                          {% endif %}
                          <form method="post" action="{% url
'add_document' %}" enctype="multipart/form-data">
                              {% csrf_token %}
                              <label for="title">Название
документа</label> <br>
```

```
                                    <input type="text" name="title"
class="form-control">
                                    <div class="mt-3">
                                        {{ imgform.docfile.label_tag }}
                                          {{ imgform.docfile }}
                                    </div>
                                    <input type="hidden" name="worker"
value="{{person.id}}">
                                    <input type="submit" value="Добавить"
class="btn btn-primary mt-2">
                                </form>
                            </div>
                            <div class="modal-footer">
                              <button type="button" class="btn btn-secondary"
data-dismiss="modal">Закрыть</button>
                            </div>
                        </div>
                      </div>
                    </div>
                </td>
                  <td>
                      {% if person.calls_expired_status.count > 0 %}
                      <a href="{% url 'student_detail' pk=person.id %}"
type="button" class="btn btn-danger">Подробнее</a>
                      {% else %}
                      <a href="{% url 'student_detail' pk=person.id %}"
type="button" class="btn btn-primary">Подробнее</a>
                      {% endif %}
                  </td>
              </tr>
              {% endfor %}
            </tbody>
          </table>

        <div class="modal fade" id="ExcelModel" tabindex="-1" role="dialog"
aria-labelledby="exampleModalLabel" aria-hidden="true">
          <div class="modal-dialog" role="document">
            <div class="modal-content">
              <div class="modal-header">
                <h5 class="modal-title" id="label">Экспорт / импорт excel</h5>
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                  <span aria-hidden="true">&times;</span>
                </button>
              </div>
              <div class="modal-body">
                  <div>
                      <p>Загрузить списком студентов (<a href="{% static
'excel/example.xls' %}" download>скачать пример</a>):</p>
                      <form action="{% url 'import_students_excel' %}"
method="post" enctype="multipart/form-data">
                          {% csrf_token %}
                          <div class="mt-3">
                          <input type="file" name="filename">
                          </div>
                          <input type="submit" value="Добавить" class="btn btn-
primary mt-2">
                      </form>

                  </div>
                  <hr>
                  <form action="{% url 'export_students_xls' %}" method="post">
                      {% csrf_token %}
```

```
                    <div class="form-check form-check-inline">
                        <label class="form-check-label" for="check_fio">ФИО</
label>
                        <input class="form-check-input" type="checkbox"
id="check_fio" value="ФИО" name="fio_check">
                    </div>
                    <div class="form-check form-check-inline">
                        <label class="form-check-label" for="check_fio">Год
выпуска</label>
                        <input class="form-check-input" type="checkbox"
id="release_year_check" value="Дата рождения" name="release_year_check">
                    </div>
                    <div class="form-check form-check-inline">
                        <label class="form-check-label"
for="check_fio">Занятость</label>
                        <input class="form-check-input" type="checkbox"
id="employment_check" value="Занятость" name="employment_check">
                    </div>
                    <div class="form-check form-check-inline">
                        <label class="form-check-label"
for="check_fio">Специальность</label>
                        <input class="form-check-input" type="checkbox"
id="specialty_check" value="Специальность" name="specialty_check">
                    </div>
                    <div class="form-check form-check-inline">
                        <label class="form-check-label" for="check_fio">Место
прохождения первой практики</label>
                        <input class="form-check-input" type="checkbox"
id="practice_one_check" value="Место прохождения первой практики"
name="practice_one_check">
                    </div>
                    <div class="form-check form-check-inline">
                        <label class="form-check-label" for="check_fio">Место
прохождения второй практики</label>
                        <input class="form-check-input" type="checkbox"
id="practice_two_check" value="Место прохождения второй практики"
name="practice_two_check">
                    </div>
                    <div class="form-check form-check-inline">
                        <label class="form-check-label"
for="check_fio">Вакансия</label>
                        <input class="form-check-input" type="checkbox"
id="vacancy_st_check" value="Вакансия" name="vacancy_st_check">
                    </div>
                    <div class="form-check form-check-inline">
                        <label class="form-check-label" for="check_fio">Вид
занятости</label>
                        <input class="form-check-input" type="checkbox"
id="on_speciality_check" value="Вид занятости" name="on_speciality_check">
                    </div>
                    <input type="submit" value="Экспорт данных в Excel"
class="btn btn-light btn-lg btn-block">
                </form>
            </div>
            <div class="modal-footer">
              <button type="button" class="btn btn-secondary" data-
dismiss="modal">Закрыть</button>
            </div>
          </div>
        </div>
      </div>
    <script>
        $(document).ready(function(){
```

```
          $("#myInput").on("keyup", function() {
            var value = $(this).val().toLowerCase();
            $("#myTable tr").filter(function() {
              $(this).toggle($(this).text().toLowerCase().indexOf(value) > -
1)
            });
          });
        });
        jQuery(function($){
        $('#code_spec').mask("99.99.99");
        $('#date_of_birth').mask("99.99.9999");
        $('#phone_number').mask("+7(999)999-9999");
        });


        $('#select_employment').on('input', function() {
          var employment = $('#select_employment option:selected').val();
          $('#employment').val(employment);
        });
    </script>
    <script>
    window.print_this = function(id) {
    var prtContent = document.getElementById(id);
    var WinPrint = window.open('', '',
'left=0,top=0,width=800,height=900,toolbar=0,scrollbars=0,status=0');
    WinPrint.document.write('<link rel="stylesheet" type="text/css"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
');
    WinPrint.document.write(prtContent.innerHTML);
    WinPrint.document.close();
    WinPrint.setTimeout(function(){
          WinPrint.focus();
          WinPrint.print();
          WinPrint.close();
        }, 1000);
    }
    </script>
    {% endblock %}
```

Листинг 13. Создаём файл settings.html для отображения страницы с настройками

```
{% extends "base_generic.html" %}

{% block title %}
<title>Настройки</title>
{% endblock %}

{% block content %}

<h3 class="text-center mt-5">Настройка цвета полей:</h3>
<form action="{% url 'settings' %}" method="post" class="container w-25 mt-5
shadow-sm p-3 mb-5 bg-white rounded">
    {% csrf_token %}
    <select name="employment" class="form-control">
        {% for employment in employments %}
        <option value="{{ employment.id }}">{{ employment }}</option>
        {% endfor %}
    </select>
    <input type="color" name="color" class="form-control mt-1">
```

```
    <input type="submit" value="Добавить" class="btn btn-primary mt-2">
</form>

<div class="card text-center mt-5 w-75 container">
  <div class="card-header">
    Список занятостей
  </div>
  <div class="card-body">
      {% for employment in employments %}
        <p class="card-text" style="background-color:
{{ employment.color }};">{{ employment }}</p>
      {% endfor %}
  </div>
</div>

{% endblock %}
```

Листинг 14. Создаём файл student_detail.html для отображения страницы студента

```
{% extends "base_generic.html" %}

{% block title %}
<title>{{ person.fio }}</title>
{% endblock %}

{% block content %}

  <div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 bg-light">
    <div class="col-md-5 mx-auto">
      <h1 class="display-4 font-weight-normal">{{ person.fio }}</h1>
      <p class="lead font-weight-normal"><p class="lead font-weight-bold">Год
выпуска:</p> {{ person.release_year }}</p>
      <p class="lead font-weight-normal"><p class="lead font-weight-
bold">Бюджет / Внебюджет:</p> {{ person.budget }}</p>
      <p class="lead font-weight-normal"><p class="lead font-weight-
bold">Место прохождения первой практики:</p> {{ person.practice_one }}</p>
      <p class="lead font-weight-normal"><p class="lead font-weight-
bold">Место прохождения второй практики:</p> {{ person.practice_two }}</p>
      <p class="lead font-weight-normal"><p class="lead font-weight-
bold">Занятость:</p> {{ person.employment }}</p>
      <a class="lead font-weight-normal"><p class="lead font-weight-
bold">Email:</p> <a href="mailto:{{ person.email }}" class="text-
reset">{{ person.email }}</a></a>
      <p class="lead font-weight-normal"><p class="lead font-weight-
bold">Номер телефона:</p> <a href="tel:{{ person.phone_number }}"
class="text-reset">{{ person.phone_number }}</a></p>
      <p class="lead font-weight-normal"><p class="lead font-weight-
bold">Образование:</p> {{ person.specialty }}</p>
    </div>
  </div>
  <hr>
  <div class="container">
    <h1 class="text-center">Текущая вакансия / добавить вакансию:</h1>
    <div class="modal-dialog" role="document">
      <div class="modal-content">
          <div class="modal-body" id="to_print_{{person.id}}">
              {% if not person.vacancy_st %}
              <h2>Умения:</h2>
              {% if person.skills_student_set.all %}
                  {% for skill in person.skills_student_set.all %}
                      <p>{{skill}}</p>
```

```
                {% endfor %}
            {% else %}
            <p>Умений пока нету :)</p>
            {% endif %}
            <h2>Добавить умение</h2>
            <form action="{% url 'student_skills' %}" method="post">
                {% csrf_token %}
                <input type="hidden" name="id" value="{{person.id}}"/>
                <input type="text" name="title" class="form-control"/>
            </form>

            {% if vacancy %}
            <h2>Вакансии:</h2>
                    {% for vac in vacancy %}
                        <h3 class="text-success">{{vac.title}}</h3>
                        <p>Умения для вакансии {{vac.title.lower}}:</p>
                        {% for skill in vac.skills_vacancy_set.all %}
                            <p>{{skill}}</p>
                        {% endfor %}
                    {% endfor %}
                        <h3>Выбор вакансии</h3>
                        <form action="{% url 'add_vacancy' %}"
method="post">
                            {% csrf_token %}
                            <input type="hidden" name="id_st"
value="{{person.id}}"/>
                            <select name="vacancy" id="vacancy"
class="form-control w-50">
                              <option value="Другое">Выбор
вакансии</option>
                                {% for vac in vacancy %}
                              <option value="{{vac.id}}">{{vac}}</option>
                                {% endfor %}
                            </select>
                            <input type="submit" value="Добавить"
class="btn btn-primary mt-2">
                        </form>
            {% endif %}
          </div>
          {% else %}
              <p>Вакансия: {{person.vacancy_st}}</p>
          {% endif %}
      </div>
    </div>
  </div>
  <hr>
  <div class="container">
      <form action="{% url 'student_detail' pk=person.id %}" method="post"
class="container mt-5">
        {% csrf_token %}
        <h2>Добавить звонок</h2>
        <input type="datetime-local" name="call_time" class="form-control mt-
1" placeholder="Дата звонка">
        <textarea name="comment" cols="30" rows="10" class="form-control mt-
3" placeholder="Комментарий к звонку (максимум 150 символов)"></textarea>
        <input type="submit" value="Добавить" class="btn btn-primary mt-2">
      </form>
  </div>
  {% if person.calls_set.count > 0 %}
    {% for call in person.calls_expired_status %}
        <div class="list-group container mb-3 mt-5">
          <div class="list-group-item active">
            <div class="d-flex w-100 justify-content-between">
```

```
              <h5 class="mb-1 text-light"><a href="tel:{{ person.phone_number
}}" class="mb-1 text-light">{{ person.phone_number }}</a>
({{call.call_time.time}})</h5>
              <small class="text-light">{{call.call_time.date}}</small>
            </div>
            <p class="mb-1">{{call.comment}}</p>
            <form action="{% url 'calls_cancle' pk=call.id %}" method="post">
              {% csrf_token %}
              <input type="hidden" value="Завершён" name="status">
              <input type="submit" value="Завершить" class="btn btn-danger
mt-2">
            </form>
          </div>
        </div>
    {% endfor %}
    <hr>
      {% for call in person.calls_not_expired_status %}
            <div class="list-group container mb-3" style="word-wrap: break-
word;">
              <div class="list-group-item">
                <div class="d-flex w-100 justify-content-between">
                  <a href="tel:{{ person.phone_number }}"><h5 class="mb-1">{{
person.phone_number }}</h5></a>
                  <small class="text-primary">{{call.call_time}}</small>
                </div>
                <p class="mb-1">{{call.comment}}</p>
              </div>
            </div>
        {% endfor %}
  {% endif %}
{% endblock %}
```

Листинг 15. Создаём файл vacancy.html для отображения страницы с вакансиями и организациями

```
{% extends "base_generic.html" %}

{% block title %}
<title>Вакансии</title>
{% endblock %}

{% block content %}
    <form action="{% url 'add_org' %}" method="POST" class="container mt-5">
        {% csrf_token %}
        <h3>Добавить организацию</h3>
        {% for field in form_org %}
            <div class="form-group row">
                <div class="col-sm-10">
                        {{ field.label }}
                        {{ field }}
                </div>
            </div>
        {% endfor %}
        <input type="submit" value="Добавить" class="btn btn-primary mt-2">
    </form>
    <hr>

    {% if org_list.count > 0 %}
            {% for org in org_list %}
                <h3 class="container">{{ org.title }}</h3>
```

```
                <div class="container mt-5 d-flex flex-wrap">
                    {% for vac in org.vacancy_set.all %}
                        {% if not vac.worker %}
                            <div class="card bg-light mb-3 ml-5"
style="width: 20rem;">
                            <div class="card-header">{{ vac }}</div>
                                <div class="card-body">
                                    <h5 class="card-title">Умения: </h5>
                                    {% for skill in
vac.skills_vacancy_set.all %}
                                        <p class="card-text">{{ skill }}</p>
                                    {% endfor %}
                                    <form action="{% url 'vacancy_skills' %}"
method="POST">
                                        {% csrf_token %}
                                        <label for="title">Добавить
умение</label>
                                        <input type="text" name="title"
class="form-control"/>
                                        <input type="hidden" name="vacancy"
value="{{ vac.id }}"/>
                                    </form>
                                </div>
                            </div>
                        {% else %}
                            <div class="card border-danger text-danger  mb-3
ml-5" style="width: 20rem;">
                                <div class="card-header">{{ vac }}
({{vac.worker.fio}})</div>
                                <div class="card-body">
                                    <h5 class="card-title">Умения: </h5>
                                    {% for skill in
vac.skills_vacancy_set.all %}
                                        <p class="card-text">{{ skill }}</p>
                                    {% endfor %}
                                    </div>
                                </div>
                            {% endif %}
                        {% endfor %}
                        </div>
                        <div class="container">
                            <button class="btn btn-primary" type="button" data-
toggle="collapse" data-target="#collapse{{org.id}}" aria-expanded="false">
                            Добавить вакансию
                            </button>
                            <div class="collapse w-25 mt-3"
id="collapse{{org.id}}" style="border: 1px solid gray; border-radius: 10px;
padding-bottom: 2em;">
                                <form action="{% url 'add_vacancy_org' %}"
method="POST" class="container mt-5">
                                    {% csrf_token %}
                                    <label for="title">Название вакансии</label>
                                    <input type="text" name="title" class="form-
control"/>
                                    <input type="hidden" name="organization"
value="{{ org.id }}"/>
                                </form>
                            </div>
                        </div>
                        <hr>
                {% endfor %}
    {% endif %}
```

```
{% endblock %}
```

Листинг 16. Создаём файл education.html для отображения страницы с университетами, специальностями и группами

```
{% extends "base_generic.html" %}

{% block title %}
<title>Образование</title>
{% endblock %}

{% block content %}
    <form action="{% url 'education' %}" method="POST" class="container mt-
5">
        {% csrf_token %}
        <h3>Добавить университет/спо</h3>
        {% for field in form %}
            <div class="form-group row">
                <div class="col-sm-10">
                        {{ field.label }}
                        {{ field }}
                </div>
            </div>
        {% endfor %}
        <input type="submit" value="Добавить" class="btn btn-primary mt-2">
    </form>
<hr>
    <form action="{% url 'add_group' %}" method="POST" class="container mt-
5">
        {% csrf_token %}
        <h3>Добавить группу</h3>
        {% for field in g_form %}
            <div class="form-group row">
                <div class="col-sm-10">
                        {{ field.label }}
                        {{ field }}
                </div>
            </div>
        {% endfor %}
        <input type="submit" value="Добавить" class="btn btn-primary mt-2">
    </form>
<hr>

    {% for college in colleges %}
      <div class="my-3 p-3 bg-white rounded box-shadow container">
        <h6 class="border-bottom border-gray pb-2 mb-0">{{ college.title }}</
h6>
          {% for spec in college.specialty_set.all %}
            <div class="media text-muted pt-3">
              <p class="media-body pb-3 mb-0 small lh-125 border-bottom
border-gray">
                <strong class="d-block text-gray-dark">{{ spec.title
}}</strong>
                  {% for group in spec.group_set.all %}
                      <strong class="d-block text-gray-
dark">{{ group.title }}</strong>
                      {% for student in group.job_set.all %}
                        <a href="{% url 'student_detail' pk=student.id %}">
{{ student.fio }} </a> <br>
                      {% endfor %}
```

```
                {% endfor %}
            </p>
          </div>
        {% endfor %}

      <small class="d-block mt-3">
          <button class="btn btn-primary" type="button" data-
toggle="collapse" data-target="#collapse{{college.id}}" aria-
expanded="false">
              Добавить специальность
          </button>
          <div class="collapse w-25 mt-3" id="collapse{{college.id}}"
style="border: 1px solid gray; border-radius: 10px; padding-bottom: 2em;">
              <form action="{% url 'specialty' %}" method="POST"
class="container mt-5">
                  {% csrf_token %}
                  <label for="title">Код и название специальности</label>
                  <input type="text" name="title" class="form-control"
placeholder="11.11.11 Специальность"/>
                  <input type="hidden" name="college"
value="{{ college.id }}"/>
              </form>
          </div>
      </small>
    </div>
  {% endfor %}
{% endblock %}
```

Приложение 2

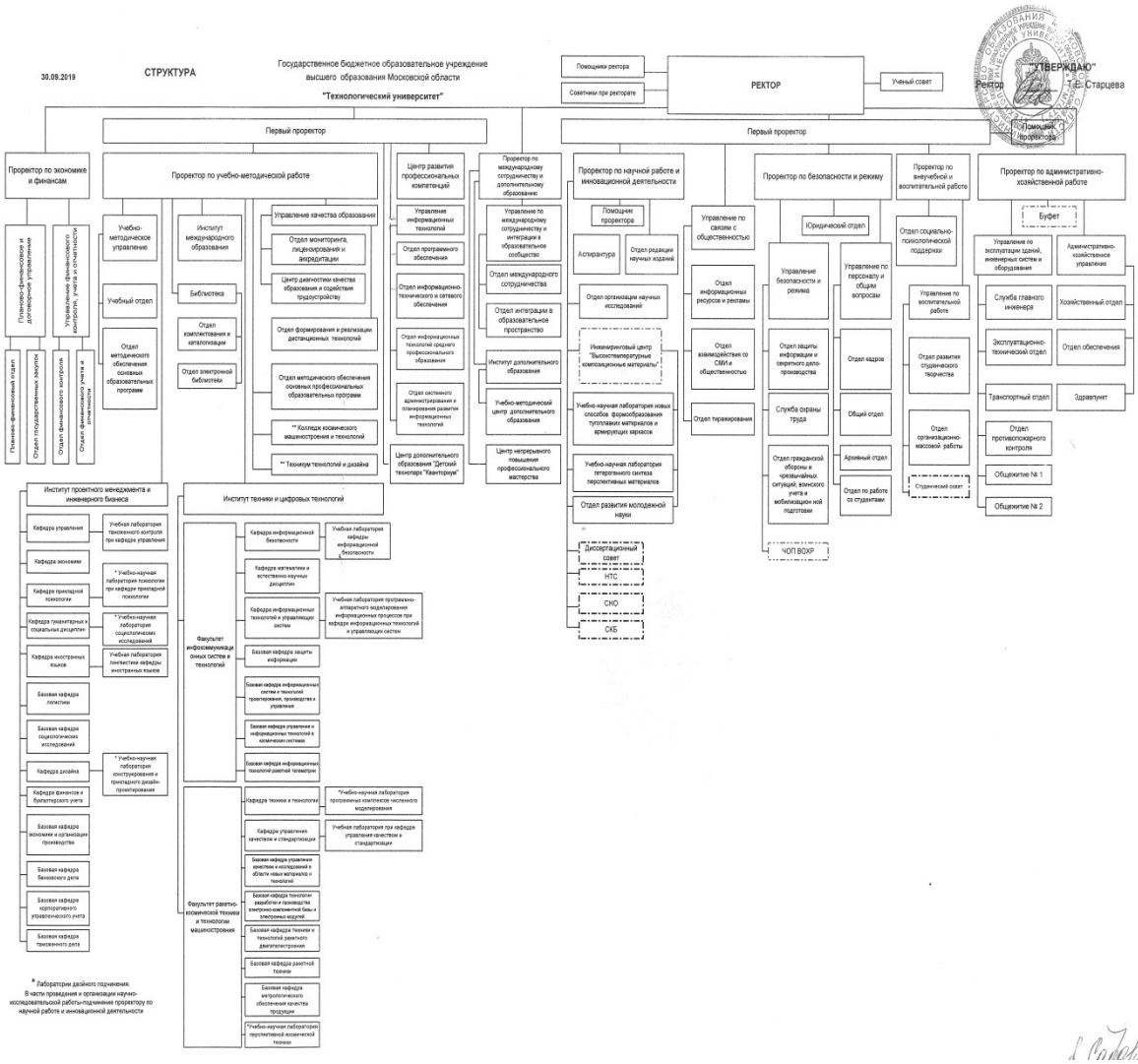

Рис. 6.1. Организационная структура «МГОТУ».

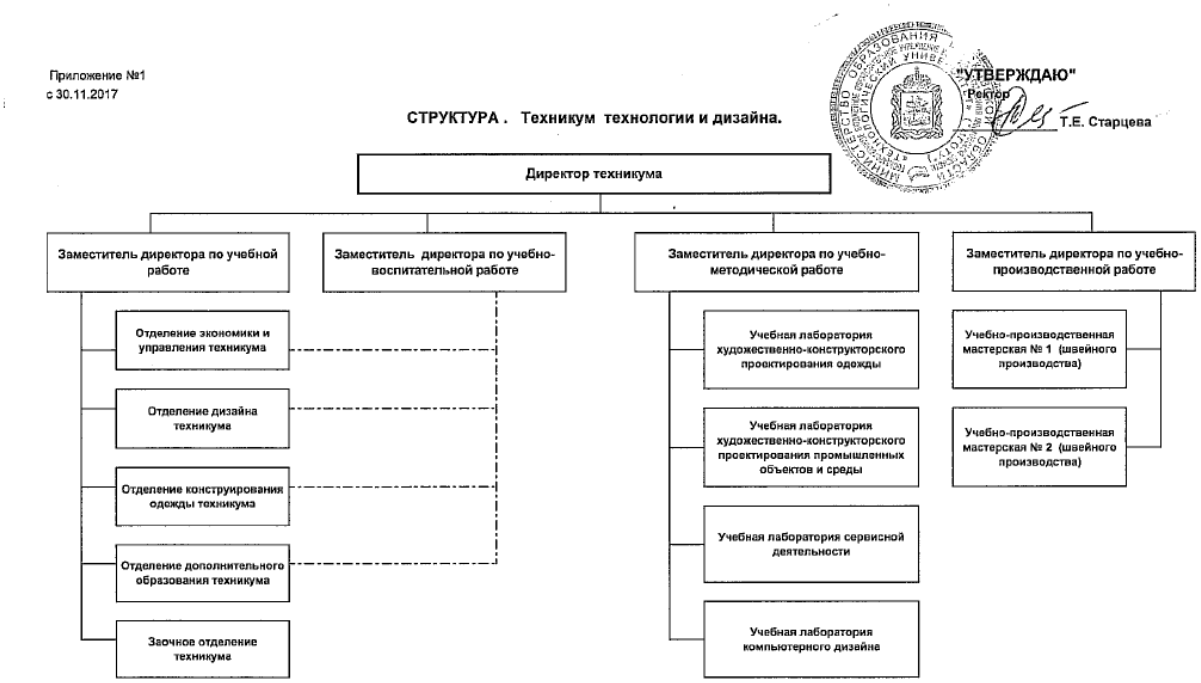

Рис.6.2 Организационная структура ККМТ