Областное государственное бюджетное профессиональное образовательное учреждение
“Рязанский Политехнический колледж”

К защите допущен

Зам. директора по УР

_____Л.В. Свечникова

ДИПЛОМНАЯ РАБОТА

по дисциплине: МДК.01.04 «Системное программирование»

на тему: Разработка базы данных «Абитуриенты» для «РПТК»

Выполнил:

студент 4 курса 30 гр.

Рудовский С.Ф.

Руководитель:

кандидат педагогических

наук, преподаватель РПТК,

доцент**,** Седов И.А.

Содержание

Введение

Обучение студентов в РПТК по специальности 09.02.07 Информационные системы и программирование позволяет выпускать специалистов с навыками программирования.

В процессе обучения данной специальности студенты приобретают такие навыки и знания как:

— разработка программных модулей программного обеспечения для компьютерных систем;

— осуществление интеграции программных модулей;

— сопровождение и обслуживание программного обеспечения компьютерных систем;

— разработка, администрирование и защита баз данных.

Для темы дипломной работы «Разработка сайта приёма абитуриентов» необходимо создать приложение-сервер, способное поддерживать работоспособный сайт, и связанную с ним базу данных для хранения информации приёмной комиссии РПТК.

Целью работы является создание серверной программы, поддерживающую работу сайта, на котором абитуриенты будут осуществлять регистрацию в РПТК и базу данных, хранящую их информацию.

Для достижения поставленной цели в работе необходимо решить следующие задачи:

1. Изучить структуру колледжа и документацию, связанную с приёмом абитуриентов;
2. Выбрать язык программирования и библиотеки, необходимые для реализации программы;
3. Спроектировать структуру базы данных для абитуриентов;
4. Разработать дизайн сайта и связать его с базой данных.

Объектом исследования в работе является структура базы данных для приёмной комиссии РПТК.

Предметом исследования являются серверные приложения на Flask.

Теоретической основой исследования являются устав Рязанского Политехнического колледжа и документы директора на тему приёма абитуриентов.

Основными теоретическими источниками являются: официальный сайт Рязанского Политехнического Колледжа, Мигель Гринберг: Разработка веб-приложений с использованием Flask на языке Python и документация библиотеки Flask.

Глава 1 Теоретические основы

1.1 Документационная основа приёма абитуриентов.

Абитуриент – выпускник средней школы. Человек, поступающий в высшее или специальное учебное заведение. В Рязанском политехническом колледже правила приёма абитуриентов, как и в других профессиональных образовательных учреждениях, регламентируется приказом Министерства образования и науки РФ.

Исходя из этого документа абитуриенту при подаче заявления необходимо также предъявить:

— Оригинал или ксерокопию документов, удостоверяющих его личность, гражданство;

— Оригинал или ксерокопию документа об образовании;

— 4 фотографии.

Также в самом заявлении поступающего указываются:

— Фамилия, имя и отчество;

— Дата рождения;

— Реквизиты документа, удостоверяющего личность;

— Данные о предыдущем уровне образования;

— Специальность, для обучения которой он планирует поступать в колледж;

— Нуждаемость в предоставлении общежития;

— Необходимость создания для поступающего спец. условий при проведении вступительных испытаний. (11)

На специальность поступают то количество абитуриентов из очереди, что указано в контрольных цифрах приёма. На 2020 год в Рязанском политехническом колледже на все специальности предусмотрено 25 бюджетных мест. Приложение А.1. (12). Очередь из абитуриентов будет сортироваться по убыванию среднего балла их аттестата.

Также по правилам приёма все абитуриенты, предоставившие в колледж ксерокопии документов об образовании и до 15 августа не передавшие оригинал аттестата, отсеиваются из очереди на специальность.

Абитуриенты, сдавшие оригиналы документов об образовании, получают расписку о документах, в которой значится, что документы переданы колледжу на хранение.

1.2 Язык программирования Python.

Python – это высокоуровневый, интерпретируемый, объектно-ориентированный и интерактивный язык программирования общего назначения, который ориентирован на повышение читаемости кода и производительности разработчика. (2)

Python это интерпретируемый язык: исходный код на Python не компилируется в машинный код, а выполняется непосредственно с помощью специальной программы-интерпретатора.

Python является интерактивным: это значит, что пользователь имеет возможность писать код прямо в оболочке интерпретатора и вводить новые команды по мере выполнения предыдущих команд.

Python это объектно-ориентированный язык программирования: он поддерживает принципы ООП, которые подразумевают инкапсуляцию кода в специальные структуры, именуемые объектами.

Также большим преимуществом языка Python является его модульность. Крупные программы на языке Python обычно организованы в виде набора модулей и пакетов. Кроме того, большое число модулей также входит в стандартную библиотеку Python. (Девид Бизли Python Подробный справочник [Текст]: Девид Бизли Пер. с англ. – СПб.: Символ-Плюс, 2010., 189).

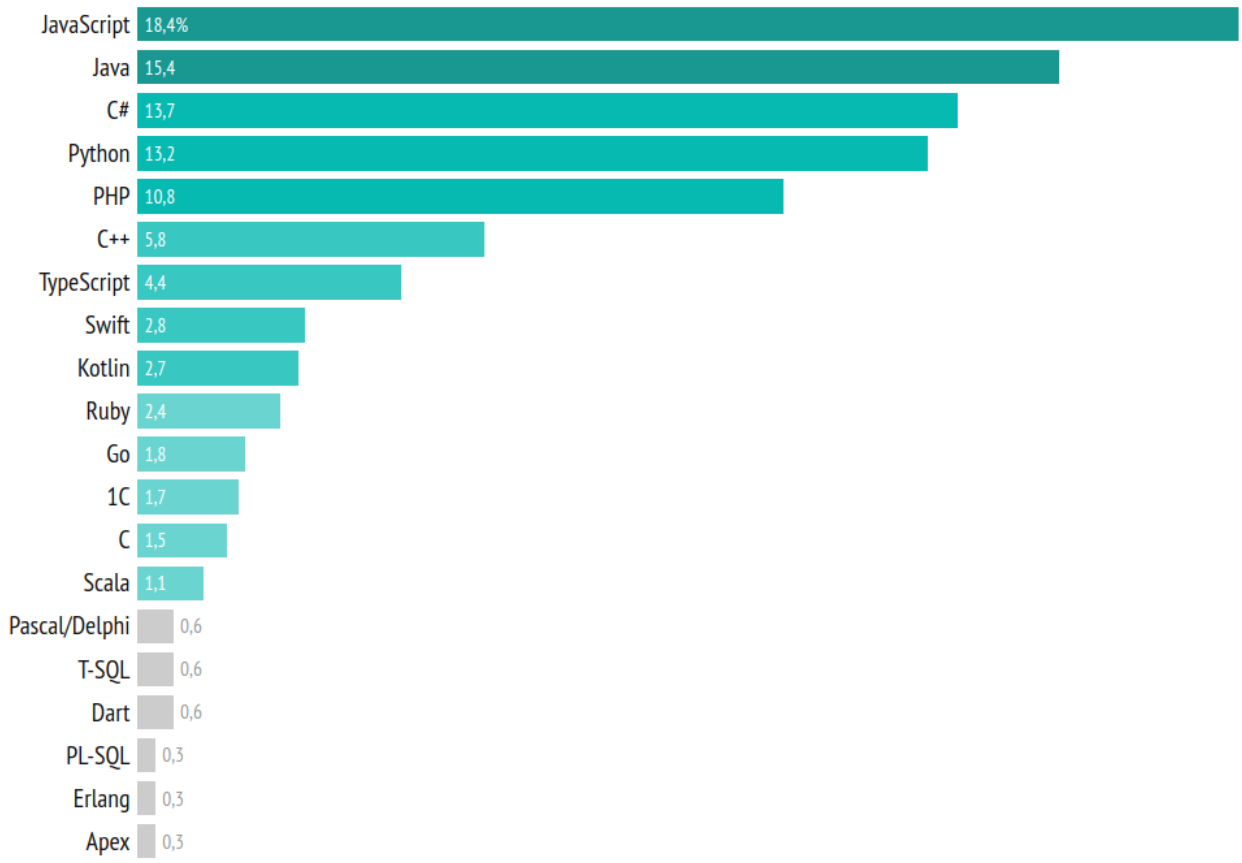
К минусам этого языка программирования можно отнести:

— Python не самый быстрый среди языков программирования. Скорость выполнения программ может быть ниже относительно других языков;

— Не самый удобный язык для мобильных разработок;

— Из-за гибкости типов данных потребление памяти Python не минимальное. (1)

В современных реалиях Python очень популярный язык программирования.



Всемирный топ языков программирования на 2020 год

И по этой причине количество прикладных библиотек для Python в самых разных областях без преувеличения огромно (веб, базы данных, обработка изображений, обработка текста, численные методы, приложения операционной системы и т. д.). (2) Python и подавляющее большинство библиотек к нему бесплатны и поставляются в исходных кодах. Более того, в отличие от многих открытых систем, лицензия никак не ограничивает использование Python в коммерческих разработках и не налагает никаких обязательств, кроме указания авторских прав.

Например, с Python поставляются такие библиотеки, как:

— «bcrypt» для работы с алгоритмами шифрования и дешифрования паролей;

— «flask» фреймворк для создания веб-приложений и сайтов;

— «re» для быстрого фильтрования строк и поиска в них, используя регулярные выражения;

— «random» для генерации случайных и псевдослучайных чисел.

Также есть огромное множество библиотек, написанных сообществом языка и размещённых на GitHub и прочих ресурсах. Популярные пакеты, внесённые в PyPI (индекс библиотек Python), можно установить с помощью встроенной в Python утилиты «pip».

На основании вышесказанного можно заключить, что Python является оптимальным языком программирования для реализации данного проекта.

1.3 Составление набора библиотек.

Из всего разнообразия стандартных и доступных для установки библиотек необходимо выбрать те, которые помогут реализовать функционал программы:

1. Создание базы данных, обработка запросов и поддержание её доступности для программы;
2. Создание серверного приложения для хостинга сайта;
3. Обработка запросов пользователей.

Для выполнения первой функции отлично подойдёт библиотека Sqlite3. SQLite – это C библиотека, реализующая легковесную дисковую базу данных (БД), не требующую отдельного серверного процесса и позволяющую получить доступ к БД с использованием языка запросов SQL. Некоторые приложения могут использовать SQLite для внутреннего хранения данных. (6). Т.к. Python поддерживает библиотеки из C, то и sqlite был перенесён на Python и добавлен в стандартные библиотеки.

Для создания серверного программного обеспечения потребуется фреймворк Flask. Данная библиотека реализует ряд сетевых возможностей, необходимых для создания web-приложения, но в тоже время оставляет множество вещей не определёнными для создания возможности выбора разработчиком необходимых ему компонентов. “Микро” в слове “микрофреймворк” означает, что Flask стремится придерживаться простого, но расширяемого ядра. Flask не будет решать за вас многие вещи, например, какую базу данных использовать. (Flask: веб-разработка капля за каплей (ru): 2016 г.)

Также для реализации защиты паролей потребуется библиотека bcrypt. Она позволяет получать хеш паролей пользователей, который будет храниться в базе данных и сравнивать вводимые пароли с хешем из базы на валидность. Данная мера необходима для защиты базы данных от взлома. Стоит сказать, что bcrypt создаёт хеши в бинарном формате, что повышает их криптостойкость.

Исходя из вышесказанного можно сделать вывод, что набор необходимых для разработки приложения библиотек определён.

1.4 Выводы теоретической части.

Из теоретической части можно заключить, что приём абитуриентов в колледж – строго регламентированное мероприятие. Для поступления абитуриенту необходимо предоставить множество документов, а приёмной комиссии удостовериться, что они оригинальные.

Среди всех библиотек стоит выделить sqlite и flask. Эти библиотеки реализуют почти весь функционал программы и чаще всего будут использоваться в коде. Первая позволяет создать и поддерживать локальную базу данных, а вторая – выводить графический интерфейс для работы с программой.

Также можно сказать, что Python – достаточно гибкий и удобный для разработчика язык программирования, поддерживающий огромное количество библиотек для выполнения практически любой задачи.

Таким образом можно назвать поставленные задачи: изучение структуры колледжа и документацию, связанную с приёмом абитуриентов выбор языка программирования и библиотек для разработки приложения; успешно выполненными.

Глава 2 Практическая часть

2.1 Разработка базы данных и подключение к серверу

База данных – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. (https://ru.wikipedia.org/wiki/База_данных)

Библиотека sqlite3 для Python предназначена для создания локальных реляционных баз данных и поддерживает выполнение инструкций по их созданию/редактированию на популярном языке SQL.

SQL – язык для формирования, манипулирования и извлечения данных из реляционной БД. Одна из причин популярности реляционныхБД в том, что, будучи правильно спроектированными, они могут оперировать гигантскими объемами данных. (Изучаем SQL. Алан Бьюли)

Для реализации нашей программы необходимо спроектировать базу данных, содержащую набор из 5 таблиц:

— abiturients;
— users;
— passports;
— claims;
— profs.

В таблице abiturients будут храниться данные о абитуриенте, а именно: его ФИО, день рождения, страна и город происхождения, а также его userID. Последнее необходимо, чтобы привязать данные абитуриента к таблице users.

В users будут храниться логины и хеши паролей всех абитуриентов, их почта и статус, который по умолчанию = «Абитуриент». Отдельно стоит сказать про метод хранения паролей. В этой таблице будут храниться не сами пароли, а их хешированный вариант в формате бинарной строки. Это позволит защитить данные других пользователей если чей-либо логин и пароль будет раскрыт. Данные строки будут генерироваться и поддерживаться отдельным модулем программы.

Таблица passport будет хранить только паспортные данные пользователей и соответствующий им id абитуриента.

Таблица claims создаётся для хранения прочей информации, связанной с подаваемым абитуриентом заявлением. Здесь будут сохранены id пользователя, id специальности, на которую он подал заявку, название школы, в которой он учился, дата её окончания и номер аттестата, а также его средний балл, место жительства, номер телефона, название военного комиссариата, иностранный язык, группа здоровья, потребность в общежитии, данные родителей и подтверждения согласий на обработку персональных данных и прочих, необходимых для создания заявления.

В таблице profs будет информация о номерах специальностей и их названиях, обучение по которым осуществляется в колледже, специальная информация, описание и путь к дополнительным фотографиям.

На этом основной список таблиц сформирован. Общая схема выглядит следующим образом. Приложение Б.1. Данная схема сделана с помощью программы DbSchema.

Для реализации данной схемы необходимо создать модуль с использованием библиотеки Sqlite3. В нём находится sql-скрипт создания таблиц и набор команд для создания базы и заполнения её таблицами. Данный модуль будет запускаться единожды при развертывании сервера. Также его можно использовать для пересоздания базы данных при появлении в ней в процессе

эксплуатации критических дефектов, но при этом все уже существующие данные будут удалены.

После создания базы достаточно будет прописать команду:

db = sqlite3.connect('database.db')

для подключения её к серверу. Это будет реализовано в отдельной функции чтобы не держать подключение к базе постоянно.

Теперь база данных полностью готова и подключена к программе. Внутри функции могут обращаться к базе данных, читать или записывать в неё информацию с помощью запросов. Можно заключить, что задача по проектированию базы данных и подключению её к программе выполнена.

2.2 Разработка серверного приложения и сайта

Для разработки сайта будет использоваться HTML, CSS и JavaScript. Этот набор необходим для придания приятного внешнего вида страницам и реализации различных возможностей по типу форм, кнопок, меню и прочего.

HTML (HyperText Markup Language, Язык гипертекстовой разметки) — язык, используемый для создания документов веб-страниц. В настоящее время используется несколько версий HTML: прочно утвер-дился HTML 4.01, а более новая и мощная, черновая спецификация HTML5 обретает популярность и получает все большую поддержку в браузерах.

HTML —язык не программирования, а разметки, он создает систему для идентификации и описания различных компонентов документа, таких как заголовки, абзацы и списки.

В то время как HTML используется, чтобы описать содержимое веб-страницы, именно каскадные таблицы стилей (Cascading Style Sheets, CSS) влияют на то, как выглядит контент.

Язык JavaScript используется для управления элементами на вебстранице, примененными к ним стилями, или даже самим браузером. Существуют и другие языки веб-сценариев, но JavaScript (также называемый ECMAScript) стандартизирован и наиболее широко распространен. (12)

Основой для разрабатываемого серверного приложения в данном случае является фреймворк Flask. Он будет реализовывать маршрутизацию страниц сайта и обработку запросов пользователей. Также данный набор библиотек, а именно Jinja2, позволяет реализовать возможность динамической генерации html-страниц, что значительно упрощает реализацию такого компонента сайта, как личный кабинет.

Структура программ с использованием Flask в основном выглядит следующим образом:

- app.py (основной код приложения)
- database.db (база данных приложения)
- templates: (папка с шаблонами html-страниц)
    - layout.html (основной макет, по которому собираются остальные страницы)
    - index.html (корневая страница сайта)
    - …
- static: (папка со статичными файлами, такими как css и картинки)
    - favicon.ico (иконка сайта)
    - mark.png
    - …

Таким образом нам необходимо создать структуру папок для дальнейшего хранения файлов. После их создания можно приступать к набору основного кода программы.

Для начала импортируем все необходимые библиотеки для создания приложения стандартным способом.

Далее в коде программы необходимо определить экземпляр класса как основу нашего приложения:

app = Flask(__name__)

Сразу после создадим и экземпляр класса для хеширования паролей из уже созданного экземпляра Flask:

bcrypt = Bcrypt(app)

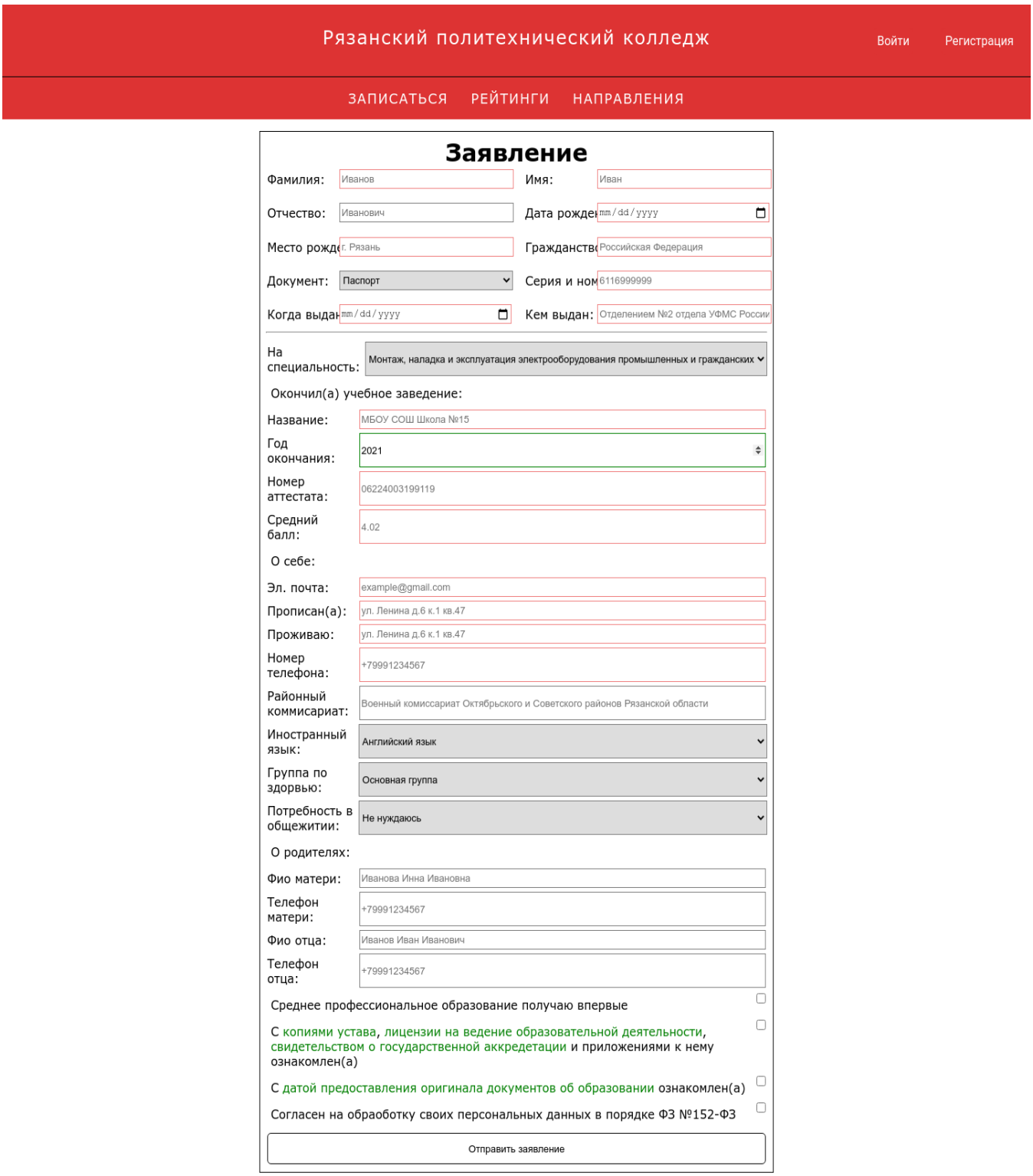
В дальнейшем это позволит буквально парой команд обрабатывать пароли пользователей.

Теперь необходимо создать несколько вспомогательных функций, которые будут отвечать за такую логику программы как: обработка запросов, проверка паролей, сохранение данных из форм и генерация pdf файлов. Они будут находиться в блоке утилит и пригодятся в дальнейшем.

Далее создадим в папке *templates* шаблон html-страницы под названием *layout.html*. Для экономии трафика css будет находиться не в отдельном файле, а прямо в коде страниц, что позволит дозированно подгружать только необходимые стили, а не всю таблицу целиком.

Суть создания шаблонов с помощью Jinja2 заключается в обозначении в теле документа блоков с помощью двойных фигурных скобок (*{{ sample }}*). Такие блоки могу нести функционал как переменных, так и циклов. Также их можно динамически заменять на другие куски HTML, которые, опять же, могут быть сгенерированы программой. Но главным инструментом Jinja2 является возможность «наследовать» одни страницы от других, что колоссально сокращает объёмы набираемого кода разработчиком. Для этого в шаблоне *layout.html* опишем только главное меню, которое будет находиться вверху страницы. Также оставим пустой *{% block body %}* для заполнения его дочерними страницами. На этом создание основного шаблона окончено.

Теперь создадим *index.html* и пронаследуем шаблон *layout.html*. Для этого в самом начале документа добавим строку *{% extends 'layout.html' %}*. Также заменяем в родительском документе блок *{% block body %}* на новый, прописав весь код страницы в нём. Таким образом два документа «соединятся» в целую страницу. Теперь при открытии главной страницы сайта пользователь вот такую страницу.



Пример главной страницы сайта

По данной методике были созданы все страницы сайта проекта.

После верстки страниц сайта необходимо прописать их маршрутизацию в основном коде программы. Для этого используется декоратор *@app.route()*. В него заворачивается функция, возвращающая строку, которая будет отправлена пользователю, выполнившему запрос по связанному адресу. Например, для маршрутизации главной страницы нужно написать следующий код:

```
@app.route(‘/’)
def index():
    retrun render_template('index.html')
```

Но данный метод актуален только для страниц, которые ничего не возвращают. А если необходимо обработать форму, заполненную пользователем, которую он отправляет на сервер, то нужно принять запрос *GET*. Для этого в декоратор необходимо добавить параметры, отвечающие за виды обрабатываемых запросов.

```
@app.route(‘/claim’, method=[‘POST’, ‘GET’])
```

Также в функцию, отвечающую за маршрутизацию данной страницы, теперь необходимо добавить обработку двух типов запросов. Для этого внутри функции разобьём обработку на два блока с if ветвлением.

```
if request.method == 'POST':
    ...
    return render_template(...)
elif  request.method == ‘GET’:
    ...
    return render_template(...)
```

Теперь при переходе на страницу *claim* пользователь увидит форму для заполнения, а при нажатии кнопки «Отправить» мы получим все данные из формы со страницы, сможем их обработать и перенаправить пользователя на другую страницу с результатом.

При помощи данных методов была разработана данная дипломная работа.

2.3 Выводы практической части

Из практической части можно заключить, что проектирование структуры базы данных – трудоёмкий процесс. Программисту необходимо сразу определиться с функционалом программы и необходимыми под его реализацию полями, и таблицами в базе данных, а также их типом и размером. Любые ошибки, внесённые в базу данных сложно исправить, а иногда проще полностью пересоздать базу данных с тестовыми значениями.

Также можно сказать, что формирование структуры интерфейса при помощи обработчика шаблонов Jinja2 значительно сокращает затраты времени на набор однотипного кода страниц и сокращает вероятность ошибок разработчика. Также шаблонизатор позволяет реализовать механизм динамической генерации страниц, что в иных случаях крайне затруднительно.

Среди всех функций стоит выделить механизм сохранения данных из формы заявления и генерацию pdf файла заявления для печати. На практике это оказались самые трудоёмкие функции по причине того, что при получении заполненной формы необходимо обработать более 30 полей и все данные занести в базу данных. После чего их необходимо в том же количестве извлечь из неё и передать в шаблон заявления для автоматизированного заполнения. При таком количестве полей есть большая вероятность разработчиком перепутать поля или что-то забыть, что при создании печатной версии заявления не критично, но при занесении данных в базу может повлечь колоссальные последствия.

Таким образом можно назвать поставленные задачи: проектирование базы данных для абитуриентов и подключение её к программе и разработка интерфейса программы; успешно выполненными.

Заключение

Систематизировав вышеприведённую информацию, можно заключить, что создание программы на языке программирования Python со своей базой данных при помощи SQLite и прочих библиотек хоть и содержит свои подводные камни, но не является невыполнимой задачей. Синтаксис языка логичен, однозначен и способствует написанию легко читаемого кода, а наличие огромного количества библиотек делает его возможности практически безграничными.

Также стоит заметить, что структура и документационное обеспечение работы приёмной комиссии в Рязанском политехническом колледже прозрачна и общедоступна. Всё документы, декларирующие этот процесс, можно найти в открытом доступе на официальном сайте колледжа. Там же можно найти и количество бюджетных мест, предоставляемое учебным учреждением на объявленные специальности. При поступлении на любую специальность от абитуриента требуется только предоставить набор документов с оригиналом аттестата. Этот процесс легко записать в базе данных программы.

Основными преимуществами программы являются её легкость в развертывании и сетевая реализация.

Первое позволяет установить сервер на любой вычислительный стенд, даже на персональный компьютер, что позволит иметь постоянный доступ к хранимым на сервере данным и его исходному коду. Это окажется незаменимой возможностью в будущем, когда появится необходимость в реализации новых функций, изменении дизайна сайта или правки/дополнения структуры базы данных сервера.

Второе позволяет не ограничиваться одним стационарным компьютером, находящимся в приёмной комиссии для подачи заявлений, а делать это и из дома, что значительно упрощает сам процесс. Но стоит отметить, что данное заявление не является окончательной формой, поскольку для зачисления на обучение требуется предоставить оригинал аттестата, что возможно при посещении

самого РПТК или отправкой заказным письмом. Если использовать второй вариант, то, при согласовании с администрацией колледжа, можно пройти процесс поступления на любую специальность в заочном формате.

Из всего текста следует, что все поставленные мною задачи выполнены и цель курсовой работы была достигнута. Мною была создана программа для приёмной комиссии Рязанского политехнического колледжа, упрощающая ведение списков абитуриентов и наглядно визуализирующая списки поступающих на специальности.

Список литературы

1. BrainSkills. Плюсы и минусы Python [Электронный ресурс]. – https://brainskills.ru/blog/plyusy-i-minusy-python/
2. Daniel Igumnov. Python [Электронный ресурс]. – https://ru.wikipedia.org/wiki/Python
3. Бизли Д. – Python. Подробный справочник
4. Вражков В. – Random Data Tools. [Электронный ресурс]. – https://randomdatatools.ru/
5. Герасимюк М. – Графический интерфейс пользователя. [Электронный ресурс]. – https://ru.wikipedia.org/wiki/Графический_интерфейс_пользователя
6. Документация Python. – sqlite3 - Интерфейс DB-API 2.0 для баз данных SQLite. [Электронный ресурс]. – https://digitology.tech/docs/python_3/library/sqlite3.html
7. Документация Python. – tkinter — Python интерфейс для Tcl/Tk. [Электронный ресурс] – https://digitology.tech/docs/python_3/library/tkinter.html
8. Макаров Л. – GUI Help/Tkinter book. [Электронный ресурс]. – https://ru.wikibooks.org/wiki/GUI_Help/Tkinter_book
9. Мирошенко Е. – База данных. [Электронный ресурс]. – https://ru.wikipedia.org/wiki/База_данных
10. Ожегов С. И. Толковый словарь [Текст]
11. Девид Бизли Python Подробный справочник [Текст]: Девид Бизли Пер. с англ. – СПб.: Символ-Плюс, 2010., 189 [Текст]
12. Дженнифер Нидерст Роббинс "HTML5, CSS3 и JavaScript. Исчерпывающее руководство". 4-ое издание [Текст]

Список используемых источников

1. Приказ Министерства образования и молодёжной политики Рязанской области – Правила приёма в Рязанский политехнический колледж в 2020 году. [Электронный ресурс]. – https://cutt.ly/DhCbTA4

2. Приложение №3 к приказу Министерства образования и молодёжной политики Рязанской области. – Контрольные цифры приёма в ОГБПОУ «Рязанский политехнический колледж» на обучение в 2020 году. [Электронный ресурс]. – https://cutt.ly/zhCbWNK

3. Мигель Гринберг: Разработка веб-приложений с использованием Flask на языке Python: https://www.labirint.ru/books/446705/

Приложение

Приложение А

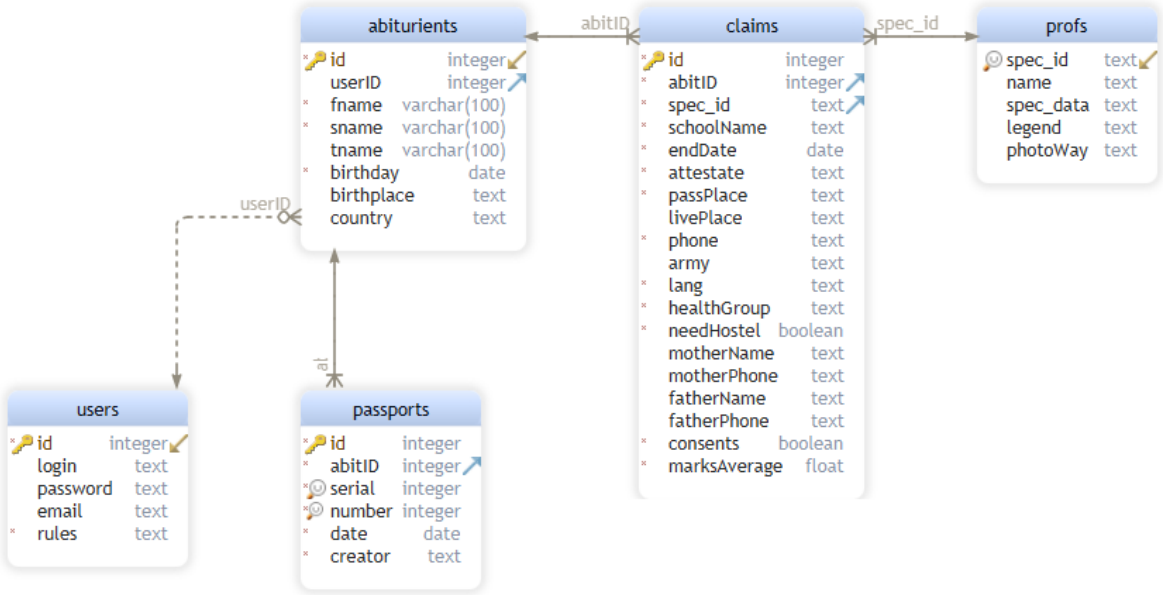
Приложение № 3
к приказу министерства образования и
молодежной политики Рязанской
от "07" 05 2020 г. № 538

Контрольные цифры приема в Областное государственное бюджетное профессиональное образовательное учреждение "Рязанский политехнический колледж" на обучение по программам среднего профессионального образования за счет средств бюджета Рязанской области на 2020 год

Наименование программ среднего профессионального образования	Код	Форма обучения		
		очная	очно-заочная (вечерняя)	заочная
Монтаж, наладка и эксплуатация электрооборудования промышленных и гражданских зданий	08.02.09	25	0	0
Информационные системы и программирование (ТОП-50)	09.02.07	25	0	0
Монтаж, техническое обслуживание и ремонт электронных приборов и устройств (ТОП-50)	11.02.16	25	0	0
Технология металлообрабатывающего производства (ТОП-50)	15.02.15	25	0	0
Эксплуатация транспортного электрооборудования и автоматики (по видам транспорта, за исключением водного)	23.02.05	25	0	0
Технология деревообработки	35.02.03	25	0	0
Экономика и бухгалтерский учет (по отраслям)	38.02.01	25	0	0
Технология эстетических услуг (ТОП-50)	43.02.12	25	0	0
Технология парикмахерского искусства (ТОП-50)	43.02.13	25	0	0
Документационное обеспечение управления и архивоведение	46.02.01	25	0	0
Монтажник радиоэлектронной аппаратуры и приборов	11.01.01	25	0	0
Оператор станков с программным управлением (ТОП-50)	15.01.32	25	0	0
ИТОГО		**300**	**0**	**0**

1. Контрольные цифры приёма в РПТК в 2020 году

Приложение Б



1. ER-диаграмма базы данных программы

Приложение В

Лицензия № 27-2574 от 26.10.2015 г.
Свидетельство об аккредитации № 27-0885 от «03» октября 2015 г.

Директору ОГБПОУ «РПТК»
Смыслову А.Ф.

Средний балл 4.18	Копия	Оригинал

Фамилия *Рудовский* Имя *Станислав* Отчество *Федорович* Дата рождения *2002-04-22* Место рождения *Рязань*	Гражданство *Российская Федерация* Документ *Паспорт* Серия *6116* № *959973* Когда и кем выдан *Отделением №2 2016-05-04*

З А Я В Л Е Н И Е

Прошу допустить меня к участию в конкурсе для поступления по профессии (специальности) *Информационные системы и программирование* на бюджетной основе по очной форме обучения в рамках контрольных цифр приёма.

О себе сообщаю следующее:

Окончил(а) в *2021* году *МБОУ СОШ Школа №15*
учебное заведение (Аттестат: *06224012681256*)
Место прописки *пр. Машиностроителей д.6 к.1 кв.47*
Место жительства *пр. Машиностроителей д.6 к.1 кв.47*
Телефоны: домашний _____ мобильный *+79155938297*
Электронная почта *rudovskii5@mail.ru*
Районный военный комиссариат *Военный коммисариат Октябрьского района*
Иностранный язык (в школе) *английский*
Группа по здоровью *основная*
Общежитие: *нуждаюсь*

Сведения о родителях:

Мать: *Рудовская Елена Евгеньевна*
+79994537216
Отец: *Рудовский Федоро Евгеньевич*
+79156198286

О себе дополнительно сообщаю:

Начальное или среднее образование получаю ***впервые*** _____ *(подпись)*

С копиями ***Устава, Лицензии*** *на ведение образовательной деятельности,* ***свидетельством о государственной аккредитации*** *и приложениями к нему, ознакомлен(а):* _____ *(подпись)*

С датой *предоставления* ***оригинала документа об образовании*** *ознакомлен(а):* ____ *(подпись)*

Согласен на обработку своих персональных данных *в порядке, установленном Федеральным законом от 27 июля 2006 г. № 152-ФЗ* ***«О персональных данных»*** *(Собрание законодательства РФ, 2006 № 31 ст. 3431):* _____ *(подпись)*

Подпись абитуриента: _____

Подпись ответственного лица приёмной комиссии: _____

«____» _____ 20 __ г

1. Пример заполненного шаблона заявления

```
############################################################
#                                                          #
#  Рудовский Станислав Федорович г. Рязань 2021 г.         #
#                                                          #
#  Программа "Приёмная комиссия для РПТК"                  #
#                                                          #
############################################################

import datetime
import os
import random
import sqlite3
import string

import pytils
from flask import Flask, redirect, render_template, request, url_for
from flask.globals import session
from flask.helpers import send_from_directory
from flask_bcrypt import Bcrypt

from config import *

app = Flask(__name__)
bcrypt = Bcrypt(app)

# settings

app.permanent_session_lifetime = datetime.timedelta(hours=1)
app.secret_key = SECURE_KEY

# /settings

# utilities


def _c(f) -> list:
    '''
    Конвертация результата запроса в массив
    '''
    r = []
    for i in f:
        r.append(i)
    return r
```

```
def req(sql: str, needCommit: bool = True) -> list:
    '''
    Обработка SQL запроса с возвращением результата и автокоммитом
    '''
    db = sqlite3.connect(DATABASE_WAY)
    c = db.cursor()
    if ';' in sql:
        temp = c.executescript(sql)
    else:
        temp = c.execute(sql)
    if needCommit:
        db.commit()
    return _c(temp)


def auth_request() -> bool:
    '''
    Проверка авторизованности пользователя в текущей сессии
    '''
    if 'userID' in session:
        sql = f"SELECT COUNT(1) FROM `users` WHERE `rowid` = '{session.get
('userID')}'"
        temp = req(sql)
        if temp[0][0] == 1:
            return True
        else:
            return False
    else:
        return False


def renderDocxTemplate(userLogin: str):
    '''
    Заполнение шаблонного заявление о конкретном пользователе
    '''
    from docx2pdf import convert
    from docxtpl import DocxTemplate

    def _check(a):
        return '' if a == None else a

    def _cHealth(a):
        if a == 'main':
            return 'основная'
        elif a == 'prepare':
            return 'подготовительная'
        elif a == 'specA':
            return 'специальная А'
        elif a == 'specB':
```

```
            return 'специальная Б'
        else:
            return ''

    def _cLang(a):
        if a == 'english':
            return 'английский'
        elif a == 'german':
            return 'немецкий'
        elif a == 'french':
            return 'французкий'
        else:
            return ''
        ...
    uID = getUserData(userLogin, 'users.id')[0]
    data = getUserData(userLogin, 'claims.marksAverage', 'abiturients.fnam
e', 'abiturients.sname', 'abiturients.tname',
                       'abiturients.birthday', 'abiturients.birthplace', '
abiturients.country',
                       'passports.serial', 'passports.number', 'passports.
creator', 'passports.date',
                       'claims.spec_id', 'claims.endDate', 'claims.schoolN
ame', 'claims.attestate',
                       'claims.passPlace', 'claims.livePlace', 'claims.pho
ne', 'users.email', 'claims.army',
                       'claims.lang', 'claims.healthGroup', 'claims.needHo
stel', 'claims.motherName',
                       'claims.motherPhone', 'claims.fatherName', 'claims.
fatherPhone')

    doc = DocxTemplate("static\\Заявление-шаблон.docx")
    specName = req(f'select name from profs where rowid = {data[11]}')[0][
0]
    context = {'avg': data[0], 'fname': data[1], 'sname': data[2], 'tname'
: data[3],
               'birthday': data[4], 'birthplace': data[5], 'country': data
[6],
               'serial': data[7], 'number': data[8], 'creator': data[9], '
date': data[10],
               'profName': specName, 'schoolDate': data[12], 'schoolName':
 data[13],
               'attestate': data[14], 'passPlace': data[15], 'livePlace':
data[16],
               'phone': data[17], 'email': _check(data[18]), 'army': _chec
k(data[19]),
               'lang': _cLang(data[20]), 'healthGroup': _cHealth(data[21])
,
               'needHostel': 'нуждаюсь' if data[22] else 'не нуждаюсь',
```

```
                'momName': data[23], 'momPhone': data[24], 'dadName': data[
25], 'dadPhone': data[26]}
    doc.render(context)
    s = 'static\\'
    name = f"Заявление-{userLogin}-{uID}"
    doc.save(s+name+".docx")
    convert(s+name+'.docx', s+name+'.pdf')
    os.remove(os.getcwd()+'\\'+s+name+'.docx')
    return name+'.pdf'


def loginPassCheck(un, pw):
    '''
    Проверка логина и пароля пользователя
    '''
    pwHash = req(f"select password from `users` where login = '{un}' ")[0]
[0]
    pwHash = pwHash[2:-1].encode('utf-8')
    if bcrypt.check_password_hash(pwHash, pw):
        return True
    else:
        False


def getUserData(user: str, *args):
    '''
    Получить информацию о пользователе по переданным аргументам
    '''
    sql = f'''select '''
    for i in args:
        sql += i + ', '
    sql = sql[:-2]
    sql += f''' from `users` join `abiturients` ON users.id = abiturients.
userID join `passports` ON passports.abitID=abiturients.id join `claims` O
N claims.abitID=abiturients.id where users.login='{user}' '''
    return req(sql, False)[0]


def saveBaseCookies(l):
    '''
    Сохранить базовые Cookie о пользователе
    '''
    temp = getUserData(l, 'users.id', 'abiturients.fname',
                       'users.rules', 'users.email', 'users.login')
    print(temp)
    session['userID'] = temp[0]
    session['userName'] = temp[1]
    session['userState'] = temp[2]
    session['userInfo'] = temp[3]
```

```
        session['userLogin'] = temp[4]


    def genNewUserLogin(sn: str) -> str:
        '''
        Генерация нового логина по фамилии
        '''
        login = pytils.translit.translify(sn)
        if req(f"select count(1) from `users` where `login` like '{login}'")[0
    ][0] == 0:
            return login
        else:
            try:
                num = int(sn[len(sn)-1])
            except ValueError:
                return genNewUserLogin(sn+'1')
            else:
                return genNewUserLogin(sn[:-1] + str(num+1))


    def genNewUserPass() -> dict:
        '''
        Генерация нового пароля
        '''
        cp = ''.join(random.choice(string.ascii_lowercase) for i in range(10))
        ph = bcrypt.generate_password_hash(cp)
        return {'password': cp, 'hash': ph}


    def saveFormData(form):
        '''
        Сохранение данных из главной формы
        '''
        login = genNewUserLogin(form.get('secondname'))
        temp = genNewUserPass()
        pswrd = temp['password']
        print('new user detected', login, pswrd)
        hash = temp['hash']
        sqlNewUser = f'''

            insert into `passports`(`abitID`, `serial`, `number`, `date`, `cre
    ator`) values(
                99999999,
                {form.get('passID')[:4]},
                {form.get('passID')[4:]},
                '{form.get('passDate')}',
                '{form.get('passCountry')}'
            );
```

```
        delete from `passports` where `abitID` = 99999999;

        insert into `users`(`login`, `password`) values(
            '{login}',"{hash}"
        );

        insert into `abiturients`(`userID`,`fname`, `sname`, `tname`, `bir
thday`, `birthplace`, `country`) values(
            (SELECT MAX(id) FROM `users`),
            '{form.get('firstname')}',
            '{form.get('secondname')}',
            '{form.get('tridname')}',
            '{form.get('birthday')}',
            '{form.get('birthplace')}',
            '{form.get('country')}'
        );

        insert into `passports`(`abitID`, `serial`, `number`, `date`, `cre
ator`) values(
            (SELECT MAX(id) FROM `abiturients`),
            {form.get('passID')[:4]},
            {form.get('passID')[4:]},
            '{form.get('passDate')}',
            '{form.get('passCountry')}'
        );

        insert into `claims`(
            `abitID`, `spec_id`, `schoolName`, `endDate`, `attestate`,
            `passPlace`, `livePlace`, `phone`, `army`, `lang`,
            `healthGroup`, `needHostel`, `motherName`, `motherPhone`,
            `fatherName`, `fatherPhone`, `consents`, `marksAverage`
        ) values(
            (SELECT MAX(id) FROM `abiturients`),
            '{form.get('specialitySelect')}',
            '{form.get('schoolName')}',
            '{form.get('schoolDate')}',
            '{form.get('schoolAttestate')}',
            '{form.get('passLivePlace')}',
            '{form.get('currentLivePlace')}',
            '{form.get('phoneNumber')}',
            '{form.get('commissariat')}',
            '{form.get('secondLanguage')}',
            '{form.get('medicalGroup')}',
            '{form.get('needHostelSelect')}',
            '{form.get('motherName')}',
            '{form.get('motherPhone')}',
            '{form.get('fatherName')}',
            '{form.get('fatherPhone')}',
```

```
            True,
            {form.get('attestateAverage')}
        );
    '''


    try:
        req(sqlNewUser)
    except Exception as e:
        if 'UNIQUE' in str(e):
            return 'Абитуриент с такими данными уже существует', login, ps
wrd
        else:
            return e, login, pswrd
    else:
        return 0, login, pswrd


def sendLoginPass(log, paw):
    '''
    Отправка логина и пароля (заглушка)
    '''
    print(log, paw)


def check_admin_pass(pwd):
    '''
    Проверка пароля администратора
    '''
    return pwd in ADMINS

# /utilities


# routing

@app.route('/new_password/<adm_pass>/<userID>')
def new_password(adm_pass, userID):
    '''
    Создание нового пароля пользователя
    '''
    if check_admin_pass(adm_pass):
        data = genNewUserPass()
        sql = f"""UPDATE `users` SET `password` = "{data['hash']}" where `
id` = {userID} """
        req(sql)
        return f"Новый пароль у {userID}: {data['password']} - {data['hash
']}"
    else:
        return redirect(url_for(404))
```

```
@app.route('/render_claim_pdf')
def render_claim_pdf():
    '''
    Маршрутизация страницы печати заявления
    '''
    if auth_request():

        import pythoncom
        pythoncom.CoInitializeEx(0)

        name = renderDocxTemplate(session.get('userLogin'))
        waybase = os.getcwd()+'\\static'
        temp = send_from_directory(waybase, name)
        return temp
    else:
        return redirect(url_for('login'))


@app.route('/login', methods=['post', 'get'])
def login():
    '''
    Маршрутизация страницы авторизации
    '''
    if request.method == 'POST':
        un = request.form.get('username')
        pw = request.form.get('password')
        if loginPassCheck(un, pw):
            saveBaseCookies(un)
            return redirect(url_for('profile'))
        else:
            return render_template('login.html', error='Wrong login or pas
sword')

    else:
        if session.get('userID') != None:
            return redirect(url_for('profile'))
        return render_template('login.html')


@app.errorhandler(404)
def page_not_found(e):
    '''
    Маршрутизация страницы 404
    '''
    return render_template('layout.html'), 404
```

```
@app.route('/cabinet')
def cabinet():
    '''
    Переадресация из кабинета в профиль пользователя (рудимент)
    '''
    return redirect(url_for('profile'))

@app.route('/claim', methods=['post', 'get'])
def claim():
    '''
    Маршрутизация формы подачи заявления
    '''
    species = req("SELECT rowid, name FROM `profs` ORDER BY `spec_id`")
    if request.method == 'POST':
        check, log, pas = saveFormData(request.form)
        if check == 0:
            saveBaseCookies(log)
            sendLoginPass(log, pas)
            session['currentSpec'] = req(
                f"select `name` from `profs` where rowid = '{request.form.
get('specialitySelect')}' ", False)[0][0]
            session['userLogin'] = log
            session['password'] = pas
            return redirect(url_for('claim_success'))
        else:
            return render_template('claim.html', species=species)
    else:
        return render_template('claim.html', species=species)


@app.route('/claim_success')
def claim_success(p=None, l=None, ps=None):
    '''
    Маршрутизация окна успешной регистрации
    '''
    if auth_request():
        # if True:
        p = session['currentSpec']
        l = session['userLogin']
        ps = session['password']
        session['password'] = None
        return render_template('claim_success.html', prof=p, login=l, pas=
ps)
    else:
        return redirect(url_for('login'))
```

```
@app.route('/profile', methods=['post', 'get'])
def profile():
    '''
    Маршрутизация в профиль пользователя
    '''
    if auth_request():
        if request.method == 'POST':
            print(request.form)
            fm = request.form

            pw = fm.get('password')
            if pw == '':
                return redirect(url_for('profile'))
            elif "'" in pw or '"' in pw or '`' in pw or ';' in pw:
                return redirect(url_for('profile', error='Wrong symbols in
 new password'))
            elif len(pw) < 6:
                return redirect(url_for('profile', error='The new password
 must be longer than 6 characters'))
            else:
                new_pw = bcrypt.generate_password_hash(pw)
                req(f"""
                    update `abiturients` set fname = "{fm.get('firstname')
}", sname = "{fm.get('secondname')}", tname = "{fm.get('tridname')}", birt
hday = "{fm.get('birthday')}" where userID = (select id from `users` where
 login = "{session['userLogin']}");
                    update `users` set login = '{fm.get('login')}', passwo
rd = "{new_pw}" where login = "{session['userLogin']}";
                """)
                session['userLogin'] = fm.get('login')
                return redirect(url_for('profile'))

        else:
            un = session.get('userLogin')
            st = session.get('userState')
            print(un, st, session.get('userID'))
            data = getUserData(un, 'abiturients.fname', 'abiturients.sname
', 'abiturients.tname', 'abiturients.birthday',
                               'passports.serial', 'passports.number', 'cl
aims.phone', 'users.login', 'users.password')
            return render_template('profile.html', username=un, state=st,
                                   userFN=data[1], userSN=data[0], userTN=
data[2], userAge=data[3], userPass=data[4]+data[5], userPhone=data[6],
                                   userLogin=data[7], userPassword=data[8]
)
    else:
        return redirect(url_for('login'))
```

```
@app.route('/logout')
def logout():
    '''
    Маршрутизация для выход из пользователя
    '''
    try:
        session.clear()
        return redirect(url_for('index'))
    except Exception as e:
        print(e)


@app.route('/ratings')
def ratings():
    '''
    Маршрутизация для отображения рейтингов пользователей
    '''
    rating = ENABLE_RATINGS
    if rating:
        temp = req("select profs.spec_id, profs.name, abiturients.fname, a
biturients.sname, claims.marksAverage, claims.abitID from profs join claim
s on claims.spec_id=profs.rowid join abiturients on claims.abitID=abiturie
nts.id order by profs.spec_id, claims.marksAverage desc ", False)
        specs = {}
        for i in temp:
            name = i[0]+' '+i[1]
            abname = i[3]+' '+i[2]
            if name in specs.keys():
                specs[name].append({'name': abname, 'avg': i[4], 'id': i[5
]})
            else:
                specs[name] = [{'name': abname, 'avg': i[4], 'id':i[5]}]
        return render_template('ratings.html', ratings=rating, species=spe
cs)
    else:
        return render_template('ratings.html', ratings=rating)


@app.route('/professions/')
def professions():
    '''
    Маршрутизация для отображения профессий
    '''
    data = req(
        "Select rowid, `spec_id`, `name`, `spec_data`, `legend`, `photoWay
` From `profs` order by `spec_id`", False)
    return render_template('professions.html', profs=data)
```

```python
@app.route('/professions/<int:id>')
def showProf(id):
    '''
    Маршрутизация для отображения конкретной профессии
    '''
    data = req(
        f" Select rowid, `spec_id`, `name`, `spec_data`, `legend`, `photoW
ay` From `profs` Where rowid = '{id}'", False)[0]
    return render_template('profPage.html', info=data, profID=id)


@app.route('/')
def index():
    '''
    Маршрутизация корневой страницы
    '''
    if auth_request():
        return redirect(url_for('profile'))
    else:
        return redirect(url_for('claim'))

# /routing


if __name__ == "__main__":
    app.debug = True
    app.run(host='0.0.0.0', port=80)
```

1. Исходный код программы