

ИТОГОВАЯ РАБОТА (ДИПЛОМ)

Тема: Разработка backend – сервиса на Python с использованием Django для менеджмента курсов обучения работе с системой 1С



2024 г.

Специальность:

Python-разработчик. Специалист

ЯТЧЕНКО ЕЛЕНА МИХАЙЛОВНА

Оглавление

[Введение 3](#_Toc169190747)

[Глава 1. Основы разработки веб-приложений 4](#_Toc169190748)

[1.1. Что такое веб-приложение 4](#_Toc169190749)

[1.2. Разработка на Python. Основы 4](#_Toc169190750)

[1.3. Фреймворк Django 4](#_Toc169190751)

[Глава 2. Описание моделей, структуры и функционала веб-приложения 5](#_Toc169190752)

[2.1. Модели приложения 5](#_Toc169190753)

[2.1.1. Категория курса (Category) 5](#_Toc169190754)

[2.1.2. Курс (Course) 5](#_Toc169190755)

[2.1.3. Студент (Student) 6](#_Toc169190756)

[2.1.4. Успеваемость студента (Performance) 8](#_Toc169190757)

[2.1.5. Оплата (Payment) 8](#_Toc169190758)

[2.1.6. Отзыв (Review) 8](#_Toc169190759)

[2.2. Структура и функционал приложения 9](#_Toc169190760)

[2.2.1. Вкладка «Главная» 9](#_Toc169190761)

[2.2.2. Вкладка «Курсы» 9](#_Toc169190762)

[2.2.3. Вкладка «Контакты» 12](#_Toc169190763)

[2.2.4. Вкладка «Личный кабинет» 12](#_Toc169190764)

[2.2.5. Раздел «Авторизация» 14](#_Toc169190765)

[2.2.6. Раздел «Обучение» 14](#_Toc169190766)

[2.2.7. Раздел «Оплата» 18](#_Toc169190767)

[2.3. Различия функционала по правам доступа. 18](#_Toc169190768)

[2.3.1. Неавторизованный пользователь. 19](#_Toc169190769)

[2.3.2. Студент 19](#_Toc169190770)

[2.3.3. Сотрудник 20](#_Toc169190771)

[2.3.4. Преподаватель 20](#_Toc169190772)

[2.3.5. Бухгалтер 21](#_Toc169190773)

[2.4. Шаблоны 21](#_Toc169190774)

# Введение

Данная работа представляет собой скорее итоговую практическую работу, чем дипломный проект, поскольку в ходе обучения условия поменялись, и я получаю всего лишь сертификат о прохождении обучения с использованием данной образовательной платформы, а не диплом о профессиональной переподготовке, как было заявлено изначально. Тем не менее, по возможности я старалась придерживаться требований к структуре и содержанию дипломного проекта касательно его практической части, но сократила количество страниц в части теории.

В первую очередь, цель данной работы – отработка практического применения полученных знаний, а также улучшение навыков самостоятельного обучения и поиска недостающей информации для реализации заявленного функционала веб-приложения.

Актуальность именно этой темы для выполнения итоговой работы заключается в том, что она имеет перспективу реального применения в организации, где я работаю в настоящее время. Одним из направлений деятельности данной организации, помимо внедрения продуктов 1С для автоматизации предприятий, являются курсы обучения работе с программой 1 С.

В рамках данного проекта разработан бэкенд-сервис на Python с использованием фреймворка Django для менеджмента курсов обучения работе с системой 1С, включающего регистрацию и учет студентов в разрезе курсов и статусов, в том числе учет данных об успеваемости и оплате. Реализована система аутентификации пользователей, а также разграничение функционала с учетом прав доступа как для студентов, имеющих различный статус (заявка на обучение, действующий студент, студент в архиве), так и для сотрудников с различными наборами прав (сотрудник, преподаватель, бухгалтер).

# Глава 1. Основы разработки веб-приложений

## Что такое веб-приложение

**Определение веб-приложения**

Веб-приложение (web-приложение) — это программное обеспечение, которое работает на удаленных серверах и доступно через интернет через веб-браузер. Оно обеспечивает пользователю доступ к различным функциям и сервисам, не требуя установки на компьютер пользователя. Веб-приложения могут выполнять широкий спектр задач, от онлайн-магазинов и социальных сетей до онлайн-банкинга и образовательных платформ.

**Веб-приложение: эволюция и сущность**

Корни веб-приложений уходят в глубину времени, когда интернет только зарождался и виртуальные сообщества формировались вокруг него. Сегодня веб-приложения стали неотъемлемой частью нашей цифровой жизни, облегчая повседневные задачи и создавая новые возможности для взаимодействия.

**Типы веб-приложений**

Существует несколько типов веб-приложений, каждый из которых предназначен для определенного типа задач и пользователей:

1. Статические веб-приложения - это веб-страницы, которые отображаются в браузере и не изменяются без перезагрузки страницы. Они часто используются для представления статического контента, такого как информация о компании или услугах.

2. Динамические веб-приложения - это приложения, которые могут изменять свое содержимое без перезагрузки страницы. Они часто используются для интерактивных приложений, таких как онлайн-магазины или социальные сети.

3. Одностраничные веб-приложения - это приложения, которые загружаются один раз и динамически обновляют содержимое при взаимодействии пользователя без перехода на другие страницы. Они часто используются для создания более быстрых и отзывчивых пользовательских интерфейсов.

**Технологии, лежащие в основе веб-приложений**

Для создания веб-приложений используются различные технологии и языки программирования, включая HTML, CSS, JavaScript, PHP, Python, Ruby и другие. Каждый из этих языков и технологий имеет свои особенности и преимущества, что позволяет разработчикам создавать разнообразные и функциональные веб-приложения.

**Примеры веб-приложений**

Существует множество примеров веб-приложений, которые мы используем ежедневно:

- Электронная почта - Gmail, Yahoo Mail, Outlook;

- Социальные сети - Facebook, Twitter, Instagram;

- Онлайн-магазины - Amazon, eBay, AliExpress;

- Образовательные платформы - Coursera, Udemy, Khan Academy;

- и прочее.

**Заключение**

Веб-приложения играют ключевую роль в нашей цифровой жизни, обеспечивая доступ к различным сервисам и упрощая повседневные задачи. Они продолжают развиваться и улучшаться, открывая новые возможности для бизнеса и общества в целом.

## Разработка на Python. Основы

### 1.2.1. Основы и Преимущества

Python - мощный и универсальный язык программирования, который применяется во многих сферах, таких как веб-разработка, анализ данных, машинное обучение и автоматизация задач. Вот несколько ключевых преимуществ Python:

1. Простота и Читаемость: Python является одним из самых читаемых и понятных языков программирования. Его синтаксис лаконичен и напоминает английский язык, что делает код более читаемым и легко поддерживаемым.

2. Широкий Функционал: Python обладает огромным количеством библиотек и модулей для решения различных задач. Благодаря этому, разработчики могут быстро и эффективно создавать программы на Python, не тратя лишнего времени на написание кода с нуля.

3. Сообщество и Поддержка: Python имеет активное сообщество разработчиков, которые постоянно работают над улучшением языка и разработкой новых инструментов. Благодаря этой поддержке, разработчики могут получить помощь, когда сталкиваются с трудностями.

### 1.2.2. Разработка на Python

Работа с веб-фреймворками

Python широко используется для создания веб-приложений благодаря мощным веб-фреймворкам, таким как Django и Flask.

- **Django** - это высокоуровневый веб-фреймворк, который предлагает множество готовых компонентов для быстрой разработки. Он обеспечивает удобство в работе с базами данных, аутентификацией пользователей и управлением URL.

- **Flask** - это легковесный фреймворк, который предлагает базовые функции, позволяя разработчикам гибко настраивать свои приложения. Он идеален для создания простых веб-приложений или API.

Использование для научных вычислений

Python популярен в научных исследованиях и анализе данных благодаря библиотекам как NumPy, SciPy и matplotlib.

- **NumPy** - предоставляет поддержку для больших многомерных массивов и матричных операций, что делает его идеальным для научных вычислений и работы с данными.

- **SciPy** - предлагает дополнительные математические функции и алгоритмы для решения научных и инженерных задач.

- **matplotlib** - используется для визуализации данных, позволяя создавать графики, диаграммы и другие визуальные представления.

Машинное обучение и искусственный интеллект

Python стал доминирующим языком для разработки в области машинного обучения и искусственного интеллекта.

- **TensorFlow** и **PyTorch** - два популярных фреймворка для глубокого обучения, которые обеспечивают высокую производительность при работе с нейронными сетями.

- **Scikit-learn** - библиотека для классического машинного обучения, предоставляющая алгоритмы для классификации, регрессии, кластеризации и других задач.

Разработка игр и графики

Python также используется в разработке игр и создании компьютерной графики благодаря библиотекам и фреймворкам, таким как Pygame и Panda3D.

- **Pygame** - простая библиотека для создания игр, обеспечивающая необходимые функции для работы с графикой, звуком и управлением взаимодействия с игроком.

- **Panda3D** - мощный фреймворк для разработки трехмерных игр и визуализации.

Работа с большими данными и облачными сервисами

Python популярен в области обработки больших данных и работе с облачными сервисами благодаря своей простоте и эффективности.

- **Apache Spark** - фреймворк для распределенной обработки данных, предоставляющий API на Python для работы с большими наборами данных.

- **AWS Lambda** и **Google Cloud Functions** - облачные сервисы, которые позволяют разработчикам создавать функции на Python для выполнения задач в облаке без необходимости управления инфраструктурой.

Это лишь небольшой обзор того, как разработка на Python применяется в различных областях. Но даже из этого краткого обзора видно, что изучение Python открывает широкие возможности для развития и карьерного роста в сфере программирования.

## Фреймворк Django

### 1.3.1. Django: Основные принципы и возможности

Django является одним из самых популярных высокоуровневых веб-фреймворков для Python, который используется для быстрой разработки веб-приложений. Вот некоторые особенности и преимущества Django:

1. Модульность и гибкость: Django предлагает множество готовых компонентов (или приложений), которые можно использовать для создания функциональных и масштабируемых веб-приложений. Это включает аутентификацию пользователей, административные панели, обработку форм, сессии и многое другое.

2. ORM (Object-Relational Mapping): Django предоставляет ORM, который позволяет работать с базами данных через объекты Python, что упрощает взаимодействие с базами данных. Django поддерживает несколько СУБД, таких как PostgreSQL, MySQL, SQLite и другие.

3. Административная панель: Одной из важных особенностей Django является встроенная административная панель, которая позволяет администраторам управлять содержимым сайта, моделями данных и пользователями, не добавляя ни строчки кода.

4. Шаблоны (Templates): Django предлагает удобный и понятный механизм для создания шаблонов, которые позволяют разработчикам создавать пользовательские интерфейсы сайтов, используя язык шаблонов Django.

5. Защита от уязвимостей: Django включает множество встроенных мер безопасности, которые защищают веб-приложения от распространенных уязвимостей, таких как XSS (межсайтовый скриптинг) и CSRF (межсайтовая подделка запросов).

### Основные шаги при работе с Django

Пример базовых шагов при использовании Django для создания простого веб-приложения:

1. Установка Django:

1. Установка Django: Для установки Django вам нужно выполнить следующую команду в командной строке (если у вас уже установлен пакетный менеджер pip): *pip install django*.

Эта команда установит последнюю версию Django на ваш компьютер.

2. Создание проекта: После установки Django вы можете создать новый проект, используя команду startproject. Например, чтобы создать проект с названием "myproject", выполните следующую команду: django-admin startproject myproject.

Это создаст директорию "myproject" с несколькими файлами, необходимыми для запуска приложения.

1. Создание приложения: Внутри проекта вы можете создать новое Django-приложение, которое будет содержать функциональность вашего веб-приложения. Выполните следующие команды для создания приложения "myapp":

cd myproject

python manage.py startapp myapp

Команда startapp создаст директорию "myapp" и несколько файлов, связанных с приложением.

4. Определение моделей: В файле models.py вашего приложения вы можете определить модели данных, которые будут использоваться в вашем веб-приложении. Модель - это класс, который представляет таблицу в базе данных. Например, вы можете создать модель Post, представляющую статьи блога, с полями title, content и date\_created.

5. Создание миграций: После определения моделей вам нужно создать миграции для применения изменений в базе данных. Выполните следующие команды: python manage.py makemigrations

python manage.py migrate

Команда makemigrations создаст файлы миграции на основе изменений в моделях, а команда migrate применит эти миграции, обновив базу данных.

6. Создание административного пользователя: Django предоставляет встроенную административную панель, которую вы можете использовать для управления данными вашего приложения. Чтобы создать учетную запись администратора, выполните следующую команду: python manage.py createsuperuser

Следуйте инструкциям в командной строке, чтобы задать имя пользователя, электронную почту и пароль для администратора.

7. Регистрация моделей в административной панели: Чтобы управлять моделями данных через административную панель, вам нужно зарегистрировать их. Для этого в файле admin.py вашего приложения добавьте код, например:

from django.contrib import admin

from .models import Post

admin.site.register(Post)

Теперь модель Post будет доступна в административной панели, и вы сможете управлять статьями блога через нее.

8. Создание представлений и шаблонов: Определите представления и шаблоны, которые определяют, как данные будут отображаться на веб-страницах. В файле views.py вашего приложения определите функции-представления, которые обрабатывают запросы и возвращают ответы. Например, вы можете создать представление post\_list, которое будет отображать список статей блога. Затем создайте соответствующий шаблон HTML, в котором будет определено, как данные будут отображаться на странице.

9. Настройка URL маршрутизации: Чтобы связать представления с URL-адресами, вам необходимо определить маршруты в файле urls.py вашего проекта. Создайте пути, указывающие на ваши представления. Например, вы можете создать путь /blog/, который будет соответствовать представлению post\_list.

10. Запуск сервера разработки: Для проверки вашего приложения вы можете запустить встроенный сервер разработки Django. Введите следующую команду:python manage.py runserver

Сервер будет запущен, и вы сможете открыть веб-браузер и перейти по адресу http://localhost:8000/, чтобы увидеть ваше веб-приложение.

Это краткий обзор процесса создания простого веб-приложения с использованием Django. Более подробная информация о каждом шаге и дополнительные возможности Django доступны в его официальной д

Это лишь краткий обзор процесса создания простого веб-приложения с использованием Django. Фреймворк предоставляет множество инструментов и функций, чтобы упростить процесс разработки и сделать ваше приложение мощным и масштабируемым.

# Глава 2. Описание моделей, структуры и функционала веб-приложения

Данное приложение представляет собой сайт для организации, которая проводит курсы обучения по программе 1С.

Ссылка на веб-приложение [здесь](https://finalcourses.pythonanywhere.com/). Кроме того, дополнительно она приложена в поле загрузки задания по курсу «Итоговая аттестация».

## 2.1. Модели приложения

В приложение включены следующие модели:

### 2.1.1. Категория курса (Category)

Поля:

\* **наименование;**

\* **флаг** категории (метка для отборов).

### 2.1.2. Курс (Course)

Поля:

\* **наименование**;

\* **категория** (внешний ключ - Category);

\* **описание** курса;

\* **продолжительность** курса, в мес.;

\* **цена** **в месяц**, азн.;

\* **дата создания**;

\* флаг "**is\_deleted**" (например, чтобы закрыть курс вместо полного удаления из базы);

\* **дата удаления** (закрытия);

\* **рейтинг** курса - недоступно для редактирования, рассчитывается автоматически при добавлении отзыва;

\* **количество просмотров** – недоступно для редактирования, регистрируется автоматически;

Кроме того, модель Курс (Course) включает **встроенные** **методы:**

**-** расчет **общей стоимости курса**;

**-** расчет **рейтинга курса**;

**-** получение URL-адреса для отображения страницы с детальной информацией конкретного курса(**get\_absolute\_url**).

### 2.1.3. Студент (Student)

Поля:

\* связанный **пользователь** (внешний ключ - User);

\* **имя**;

\* **фамилия**;

\* **номер** **телефона**;

\* **дата рождения**;

\* **курс** (внешний ключ - Course);

\* **фото**;

\* **статус** - выбор из набора значений, устанавливается автоматически;

\* **дата запроса** (подачи заявки) – автоматически, не редактируется;

\* **дата регистрации** (подтверждение заявки, начало обучения) – автоматически, не редактируется;

\* **дата изменения** (при внесении изменений в запись) – автоматически, не редактируется;

\* флаг "**is\_deleted**" (деактивация вместо полного удаления из базы);

\* **дата удаления** (деактивации);

**Статусы** устанавливаются автоматически:

- при подаче заявки – requested;

- при подтверждении заявки - approved (кроме того, происходит автоматическое добавление в группу Students);

- при подтверждении завершения обучения - finished (параллельно происходит автоматическое удаление из группы Students и перевод в группу Archive).

Кроме того, модель Студент (Student) включает встроенные **методы:**

- **полное имя** (@property);

**-** расчет **среднего балла** студента;

**-** расчет **остатка к оплате**;

**-** получение URL-адреса для отображения страницы с детальной информацией конкретного курса(**get\_absolute\_url**);

- **признак** **завершения обучения** (@property).

**Остаток к оплате** рассчитывается с учетом общей стоимости курса и уже зарегистрированных платежей от данного студента по данному курсу. Выводится для справки при заполнении формы оплаты (чтобы сотрудник видел оставшуюся задолженность студента по оплате курса), а также используется при валидации суммы оплаты при обработке POST-запроса.

**Признак** **завершения обучения** (is\_finished)рассчитывается с учетом даты регистрации студента на курсе и продолжительности курса. Студенты, у которых этот признак равен Истине, выводятся на панели навигации отдельным списком (видимость только для сотрудников) с возможностью добавить их в архив (по кнопке "Добавить в архив"), когда уже изменяется статус студента и группа пользователя, к которому он привязан (если для этого пользователя нет других активных студентов).

Т.е. предусмотрена возможность ручного подтверждения завершения обучения, т.к. могут быть незакрытые вопросы по документам, оплате и проч. Но при этом сотрудник знает, что период обучения студента завершен.

### 2.1.4. Успеваемость студента (Performance)

Поля:

\* **оценка** - выбор из набора значений;

\* **студент** (внешний ключ - Student);

\* **дата оценки;**

\* **дата изменения** оценки (предусмотрена, т.к. в перспективе доработка возможности корректировки оценок).

### 2.1.5. Оплата (Payment)

Поля:

\* **студент** (внешний ключ - Student);

\* **сумма** оплаты;

\* **дата** **оплаты** (каким числом датирован документ проведения оплаты);

\* **документ**, подтверждающий оплату - файл (с выведением по ссылке в личном кабинете и на странице детальной информации о студенте)

\* **дата добавления;**

\* **дата изменения;**

Предусмотрено внесение оплаты по частям.

В формах добавлены дополнительные методы валидации (clean) данных:

- контроль, чтобы дата оплаты не превышала текущую дату;

- контроль, чтобы сумма оплаты не была отрицательной или равной нулю;

- проверка, что общая сумма оплаты не превышает задолженность студента.

### 2.1.6. Отзыв (Review)

Поля:

\* **студент** (внешний ключ - Student), который оставил отзыв;

\* **курс** (внешний ключ - Course), к которому относится отзыв;

\* **текст** отзыва;

\* **оценка** - выбор из набора значений;

\* **дата добавления**;

Модель Отзыв (Review) включает также встроенный **метод выведения только части содержимого** текста.

Причем, отзывы на курс оставлять могут только студенты, которые обучаются или закончили обучение на данном курсе.

Сотрудники, а также прочие пользователи (даже в случае, если они подали заявку на выбранный курс, но еще не переведены в статус действующих студентов), оставлять или редактировать отзывы не могут.

## 2.2. Структура и функционал приложения

### 2.2.1. Вкладка «Главная»

При переходе на вкладку отображается главная страница, где в настоящее время представлена просто общая вводная информация: приветствие и причины для изучения 1 С, в дальнейшем планируется внести некоторые изменения. Например, добавить карусель из 5 наиболее популярных курсов (отбор по количеству просмотров).

Большая часть прочих шаблонов страниц наследуют от главной страницы (base.html).

### 2.2.2. Вкладка «Курсы»

Вкладка при нажатии отображает доступные Категории курсов. При выборе категории открывается страница со списком курсов **выбранной категории**. Для каждого курса в списке показано Наименование, Продолжительность обучения, а также кнопка "Добавить оценки по курсу".

Кнопка "Добавить оценки по курсу" перенаправляет на страницу с формой для добавления оценок. Причем, к выбору доступны только **студенты выбранного курса**. После добавления оценки одному студенту, форма повторно открывается для добавления оценки другому студенту этого же курса.

Чтобы завершить работу с формой и вернуться на страницу списка курсов, пользователю необходимо нажать кнопку "Завершить".

На странице со списком курсов «Наименование курса» является активной ссылкой, которая открывает страницу с детальной информацией по выбранному курсу. А именно:

#### 2.2.2.1. Общая информация по курсу:

- **Наименование**;

- **Категория**;

- Краткое **описание**;

- **Продолжительность** обучения;

- **Полная стоимость** обучения;

- **Рейтинг** (обновляется автоматически при добавлении нового отзыва о курсе);

- **Количество просмотров**;

- кнопка "**Подать заявку**".

При нажатии на кнопку «Подать заявку» пользователь (если он уже зарегистрирован и аутентифицирован) перенаправляется на форму регистрации нового студента. В противном случае, идет перенаправление на страницу аутентификации/регистрации на сайте.

При **подаче заявки** добавлена **проверка**: если для этого пользователя на данном курсе уже зарегистрирован студент со статусом "approved" (активный студент) либо со статусом "requested" (подана заявка на обучение), выводится соответствующее сообщение и заявка не регистрируется. Если же нет, то откроется форма для регистрации нового студента.

При этом в форме Имя, Фамилия и Электронная почта заполнены по умолчанию (с возможностью изменения) соответствующими данными текущего пользователя.

**В результате** успешной отправки и валидации формы создается экземпляр модели Студент с указанными данными, статус по умолчанию устанавливается 'r' (requested) и происходит перенаправление в личный кабинет пользователя.

#### 2.2.2.2. Отзывы:

- **Форма для добавления нового отзыва** (доступна ТОЛЬКО для студентов, которые прошли или проходят обучение на данном курсе). Форма включает в себя поля:

\* комментарий;

\* оценка (от 1 до 5).

При добавлении нового отзыва автоматически пересчитывается Рейтинг курса

- **Список** уже добавленных **комментариев**. Отображается следующая информация:

\* пользователь;

\* дата добавления;

\* оценка;

\* текст комментария.

#### 2.2.2.3. Студенты

Представлен список действующих (статус 'а' - approved) студентов данного курса с указание ФИО и среднего балла студента.

Если на данном курсе в настоящее время действующих студентов нет, выводится соответствующее сообщение.

ФИО студента является активной ссылкой, которая ведет на страницу с детальной информацией по данному студенту. Описание детальной страницы студента будет дано ниже по тексту (п.).

### 2.2.3. Вкладка «Контакты»

Добавлена вымышленная информация - контакты организации (адрес, телефон, адрес электронной почты контактного лица).

### 2.2.4. Вкладка «Личный кабинет»

Открывается личный кабинет пользователя с детальной информацией студента. Если пользователь обучается/ обучался/ подал заявку на обучение более, чем на одном курсе, вся информация будет предоставлена в разрезе курсов.

В личном кабинете отражаются следующие данные:

- **фото**;

- кнопка "**Добавить фото**" (если фото отсутствует) - переводит на форму добавления фото;

- ссылка "**Редактировать фото**" (если фото есть) переводит на форму добавления фото, фото заменяется в базе данных, а не добавляется новая запись;

- **телефон**;

- **электронная почта**;

- **дата рождения**.

Для тех, кто еще только подал заявку на обучение, но не подтвержден:

- **дата подачи заявки**;

- сообщение, что заявка находится на рассмотрении.

Для активных студентов и студентов, находящихся в архиве:

- **дата регистрации**;

- **средний балл** (успеваемость) - пересчитывается автоматически при добавлении новых оценок;

- пометка "**Обучение завершено**" (только для студентов в архиве);

- **статус оплаты** (оплачено частично, оплачено полностью, оплаты нет) - данный статус автоматически определяется (и изменяется) при добавлении платежей;

- **блок** "**Данные оплаты**", где отражается такая информация как:

\* **общая стоимость** обучения;

\* **остаток для оплаты**;

\* **данные по платежам** (если платежей нет - соответствующее сообщение):

- сумма;

- дата оплаты;

- документ, подтверждающий оплату (ссылка). Если документ (файл) отсутствует, выводится соответствующее сообщение;

Помимо вкладок, на боковой панели навигации представлены следующие разделы:

### 2.2.5. Раздел «Авторизация»

- Имя пользователя;

- Ссылки на Вход / Выход

По ссылке "Вход" идет переадресация на форму аутентификации пользователя (логин, пароль). Там же есть ссылка для возможности регистрации, если пользователь еще не зарегистрирован на сайте. В форме регистрации добавлено поле подтверждения пароля и верификация на совпадение пароля и его подтверждения.

По ссылке "Выход" производится выход и перенаправление на главную страницу сайта.

### 2.2.6. Раздел «Обучение»

Видимость данного раздела - только для сотрудников (is\_stuff).

Раздел содержит 3 ссылки:

- Заявки на обучение (с указанием количества, если заявки есть);

- Студенты;

- Архив.

Каждая из ссылок ведет на страницу со списком студентов соответствующего статуса (required, approved, finished).

#### 2.2.6.1. Заявки на обучение

Происходит перенаправление на страницу, которая **в разрезе курсов** отображает список людей, которые подали заявки на обучение (статус = required).

Указывается Курс, фамилия и имя студента, дата подачи заявки.

Тут же выводится **кнопка "Обработать заявку"**, которая перенаправляет на форму подтверждения заявки.

При выборе варианта "**Принять**" происходит:

**1)** изменение статуса студента на 'approved';

**2)** устанавливается текущая дата в качестве даты регистрации;

**3)** пользователь добавляется в группу Students.

Вариант «**Отклонить**» временно просто возвращает на предыдущую страницу (request.GET["next"]), планируется доработка что именно делать в этом случае. Как вариант - деактивация/удаление студента (is\_deleted=True).

#### 2.2.6.2. Студенты

По ссылке происходит переход на страницу, которая **в разрезе курсов** отображает список активных студентов (статус = approved).

Для каждого студента указывается курс, фамилия и имя студента, дата регистрации.

Тут же для студентов, период обучения которых завершился на текущую дату (проверка по дате регистрации и продолжительности курса) выводится соответствующее сообщение и **кнопка "Добавить в архив"**, которая перенаправляет на форму подтверждения добавления в архив.

При выборе варианта "**Принять**" происходит:

1) изменение статуса студента на 'finished';

2) пользователь удаляется из группы Students и добавляется в группу Archived (ТОЛЬКО в случае, если с данным пользователем нет связанных студентов, которые не являются активными студентами на других курсах. Иначе группа остается Students).

Вариант «Отклонить» временно просто возвращает на предыдущую страницу (request.GET["next"]), планируется доработка что именно делать в этом случае. Как вариант - деактивация/удаление студента (is\_deleted=True).

#### 2.2.6.3. Архив

Ссылка переводит на страницу, на которой **в разрезе курсов** выводится список студентов, закончивших обучение (статус = finished). Для каждого студента указывается курс, фамилия и имя студента.

Фамилия и имя студента во всех вышеуказанных ссылках раздела "Обучение", являются активными ссылками и ведут на страницу с **детальной информацией студента**. Наполнение данной страницы имеет некоторые различия в зависимости от статуса студента, что описано ниже.

**Детальная информация студента**

Отображается сходная информация, как в личном кабинете, но с небольшими различиями доступного функционала.

- **фото** (если есть);

- **номер телефона**;

- **электронная почта**;

- **дата рождения**;

- **курс**, на котором обучается/обучался/подал заявку на обучение студент;

Для тех, кто еще только подал заявку на обучение, но не подтвержден (статус = required):

- **дата подачи заявки**;

- кнопка "**Обработать заявку**" - ссылка на форму Подтверждения заявки (аналогично одноименной кнопке в списке "Заявки на обучение").

Для активных студентов и студентов, находящихся в архиве (статус = approved и статус = finished):

- **дата регистрации**;

- **средний балл** (успеваемость) - пересчитывается автоматически при добавлении новых оценок;

- кнопка "**Добавить в архив**" (для студентов, у которых период обучения завершился, но они еще не переведены в архив) - перенаправляет на форму подтверждения добавления в архив (аналогично одноименной кнопке в списке "Студенты").

- пометка "**Обучение завершено**" (только для студентов в архиве);

- **статус оплаты** (оплачено частично, оплачено полностью, оплаты нет) - данный статус автоматически переопределяется при добавлении/редактировании платежей;

- кнопка "**Добавить данные по оплате**" - открывает форму добавления данных по оплате данного студента. Кнопка не отображается, если значение статуса оплаты – «Оплачено полностью».

При этом в форме по умолчанию заполнено ФИО текущего студента, сумма оплаты (равная неоплаченному остатку), текущая дата;

При отправке формы **дополнительно проверяется сумма** (не должна быть меньше или равной нулю и не должна превышать оставшийся неоплаченный остаток) и **дата** платежа (не должна превышать текущую дату).

- блок "**Данные оплаты**", где отражается информация:

\* общая стоимость обучения;

\* остаток для оплаты;

\* данные по платежам (если платежей нет - соответствующее сообщение): сумма, дата оплаты, документ, подтверждающий оплату (ссылка). Если документ (файл) отсутствует, выводится соответствующее сообщение;

- кнопка "**Редактировать платеж**" - открывает форму добавления платежа, с той лишь разницей, что по умолчанию поля заполнены данными платежа, который хотим редактировать (студент, дата платежа, сумма).

Проверки при отправке формы срабатывают такие же, как и при добавлении платежа. Но при расчете остатка к оплате учитывается тот факт, что платеж не добавляется новый, а изменяется уже существующий.

### 2.2.7. Раздел «Оплата»

Видимость данного раздела, как и кнопок добавления/редактирования платежей, описанных выше, - только для пользователей группы **Accountant** (т.е. пользователей с правами на добавление/изменение платежей).

Раздел содержит одну ссылку - **"Добавить платеж"**. Кнопка открывает форму добавления платежа, где к выбору доступны все студенты. При этом в форме по умолчанию заполнена сумма оплаты (равная неоплаченному остатку) и текущая дата.

Кроме того, в форме есть поле ввода файла - документа оплаты.

При отправке формы дополнительно проверяется сумма (не должна превышать неоплаченный остаток и не должна быть отрицательной или нулевой) и дата платежа (не должна превышать текущую дату).

## 2.3. Различия функционала по правам доступа.

В настоящее время в приложении применяется разделение пользователей на **6 групп** (с различием по доступам к функционалу сайта):

1) **Неавторизованные** пользователи;

Авторизованные пользователи:

2) **Студенты** (группа Students, группа Archived) - права на добавление/изменение модели Review);

Разделение сделано на перспективу, для разграничения прав доступа.

4) **Сотрудники** (признак is\_stuff);

5) **Преподаватели** (признак is\_stuff, группа Teachers) - права на добавление/изменение модели Performance;

6) **Бухгалтеры** (признак is\_stuff, группа Accountants) - права на добавление/изменение модели Payment.

Ниже чуть подробнее представляю описание доступных действий/просмотра для представителей каждой из групп.

### 2.3.1. Неавторизованный пользователь.

Неавторизованному пользователю доступны вкладки "Главная", "Курсы", "Контакты" и раздел "Авторизация" боковой панели.

#### 2.3.1.1. Вкладки «Главная», «Контакты»

Ограничений нет.

#### 2.3.1.2. Вкладка «Курсы»

Доступен список курсов (выбранной Категории), по которым предоставляется обучение. Для каждого курса отображается название и продолжительность.

Не отображается кнопка "Добавить оценки по курсу".

При клике на курс открывается его детальная информация, где видно описание курса, продолжительность, стоимость, рейтинг, отзывы. Тут же кнопка "**Подать заявку**", которая сначала перенаправит пользователя на страницу аутентификации (либо регистрации на сайте по ссылке на той же странице), и после аутентификации пользователь может перейти к выбранному курсу и подать заявку на обучение.

Не отображается Форма добавления отзыва, Список студентов курса

### 2.3.2. Студент

Студенту доступны те же вкладки "Главная", "Курсы", "Контакты" и раздел "Авторизация". Помимо этого, отображается еще и вкладка "Личный кабинет".

#### 2.3.2.1. Вкладки «Главная», «Контакты», «Личный кабинет»

Ограничений нет.

#### 2.3.2.2. Вкладка «Курсы»

Аналогично неавторизованному пользователю, доступен список курсов, по которым предоставляется обучение. Для каждого курса отображается название и продолжительность.

Не отображается кнопка "Добавить оценки по курсу".

На странице детальной информации по курсу добавляется возможность оставить отзыв о курсе (форма для добавления отзыва).

Не отображается Список студентов курса

### 2.3.3. Сотрудник

Помимо вкладок и разделов, описанных выше для студента, становится доступным раздел "Обучение" - Заявки на обучение, Студенты, Архив.

#### 2.3.3.1. Вкладки «Главная», «Контакты», «Личный кабинет»

Ограничений нет (хотя в личном кабинете нет информации для сотрудников, пока там предусмотрена информация только для студентов).

#### 2.3.3.2. Вкладка «Курсы»

Аналогично доступен список курсов, по которым предоставляется обучение. Для каждого курса отображается название и продолжительность.

Не отображается кнопка "Добавить оценки по курсу".

На странице детальной информации по курсу доступны:

- блок общей информации о курсе (краткое описание, продолжительность, полная цена, рейтинг);

- отзывы по курсу (не отображается Форма добавления отзыва);

- список студентов курса.

В списке студентов курса ФИО студента является кликабельной ссылкой, которая переводит на страницу с детальной информацией по студенту. Здесь доступен весь функционал, за исключением возможности добавлять/редактировать платежи. Не отображаются кнопки "Добавить данные по оплате", "Редактировать платеж".

Также недоступным остается раздел "Оплата" на боковой панели навигации, как и для Студентов и Неавторизованных пользователей.

### 2.3.4. Преподаватель

Набор доступов сходен с Сотрудником.

Отличие в том, что на странице списка курсов для преподавателей также есть возможность выставить оценки студентам того или иного курса - кнопка "Добавить оценки по курсу"

### 2.3.5. Бухгалтер

Набор доступов сходен с Сотрудником.

Отличия:

- на боковой панели отображается раздел "Оплата" со ссылкой "Добавить платеж"

- на странице с детальной информацией студента есть кнопки "Добавить данные по оплате" и "Редактировать платеж".

Помимо уже реализованного, планируется дальнейшая доработка и расширение функционала, как и было сказано выше.

## 2.4. Шаблоны

Веб-приложение включает в себя следующие шаблоны:

- шаблон базовой/главной страницы, от которой наследуют все прочие шаблоны (base.html);

- шаблон страницы контактов (contacts.html);

- шаблон страницы личного кабинета (profile.html);

- шаблон страницы со списком курсов выбранной категории (courses\_list.html);

- шаблон страницы с детальной информацией по курсу, включает в том числе форму добавления отзыва (course\_detail.html);

- шаблон страницы со списком студентов (students\_list.html) - переход на страницу происходит со всех трех ссылок раздела «Обучение» (Заявки на обучение, Студенты, Архив), но с различиями по содержимому, в зависимости от переданного в контекст статуса студентов;

- шаблон страницы с детальной информацией студента (student\_detail.html);

- шаблон с формой подтверждения выбора (confirmation.html) - применяется как для подтверждения заявки на обучение, так и для подтверждения добавления студента в архив;

- шаблон с формой для входа в аккаунт (login.html);

- шаблон страницы с информацией об успешном выходе из аккаунта (logged\_out.html);

- шаблон с формой регистрации нового пользователя (register.html);

- шаблон страницы с сообщением об успешном завершении регистрации пользователя и переходом на форму входа в аккаунт (register\_done.html);

- шаблон с формой подачи заявки на обучение (student\_form.html);

- шаблон с формой добавления/изменения фото студента, с выведением фото на страницу при успешном его добавлении (photo.html);

- шаблон с формой для добавления/редактирования платежей (payment\_form.html);

- шаблон с формой для добавления оценок студентов (mark\_form.html).

Модели: студент, категория, курс, оплата, оценка, Отзыв

суперюзер - full\_admin, 123456123

username - finalcourses

email - на mail.ru

password - Леголас (к базе данных тоже)

admin full\_admin, admin123456

users 111user111 и т.п.

ссылка на сайт - finalcourses.pythonanywhere.com