ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки бакалавров по направлению   
01.03.02 «Прикладная математика и информатика»**

*Селивановская Дарья Евгеньевна*

**КУРСОВАЯ РАБОТА**

Разработка серверного web-приложения используя современные фрэймворки.

|  |  |
| --- | --- |
| Рецензент  д-р …. наук, проф.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  И.О. Фамилия | Научный руководитель  д-р …. наук, проф.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  И.О. Фамилия  Консультант (при наличии)  д-р …. наук, проф.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  И.О. Фамилия |

Нижний Новгород, 2020

# План работы

1. **Введение**
   1. Обоснование выбора темы курсовой работы
   2. Актуальность выбранной темы
2. **Цели**
3. **Задачи**
4. **Основные термины**
5. **Обзор языков программирования**
   1. Python
   2. Java
   3. PHP
6. **Обзор фреймворков**
   1. Django
   2. Spring
   3. Vue, React, Angular
7. **Обзор СУБД**
   1. SQLite
   2. MySQL
   3. PostgreSQL
8. **Front-end**
9. **Техническое описание приложения**
   1. Структура проекта на Django
   2. Структура приложений
   3. Django Models
   4. Работа с формами
   5. Сессии
   6. Возможности пользователя панели администратора
10. **Заключение**
11. **Список литературы и ссылок**

# Введение

# 1.1. Обоснование выбора темы курсовой работы

Современность не перестаёт удивлять разнообразием инструментов получения и обработки информации. Одним из таких инструментов является веб-сайт. Это - визитная карточка любой уважающей себя компании и способ общения с клиентами. Веб-ресурс помогает донести информацию максимально быстро до огромного количества людей, дает возможность сбора данных о пользователях и обеспечивает оперативную связь с филиалами и представителями в разных регионах страны и за рубежом. Мне всегда было интересно узнать о разработке веб-приложений “изнутри”, поэтому мной и была выбрана данная тема курсовой работы.

# 1.2. Актуальность выбранной темы

Системы работы веб-ресурсов актуальны для изучения потому что их появление произвело грандиозные революционные изменения в сферах образования, коммуникации, рынке товаров и услуг, компьютерных тренингах, в науке, искусстве и т.д. В наше время мы можем экономить массу времени и энергии оплачивая счета через интернет-порталы, занимаясь самообразованием онлайн и даже покупая необходимые товары в интернет-магазинах. Возможности интернета безграничны. А главный способ взаимодействия пользователя с этой огромной сетью – это веб-сайты и приложения. В связи с этим, необходимо изучать и развивать сферу веб-разработки, помогая пользователям быстрее получать нужную информацию и открывая новые возможности всемирной сети.

# Цели работы

1. Разработать серверное web-приложение для помощи студентам первого курса в изучении основ математического анализа.
2. Углубить знания, полученные в результате изучения курса веб-программирования (HTML, CSS, JavaScript).
3. Улучшить навыки разработки на языке Python.
4. Изучить новые средства разработки (Django).
5. Получить навыки работы с базами данных.
6. Получить знания о работе web-сайтов и web-серверов.
7. Улучшить навыки поиска и обработки информации в процессе разработки web-приложения.

# Задачи

Для достижения вышеприведенных целей я составила ряд задач:

1. Подготовить контент для приложения.
2. Разработать структуру проекта.
3. Разработать приложения (apps).
4. Написать frontend-представление для пользователей.
5. Обеспечить возможность сбора статистики по приложению.
6. Собрать и выгрузить веб-приложение на сервер.

# Основные термины

**Фреймворк** (англ. **Framework**) — [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), облегчающее разработку и объединение разных компонентов большого программного проекта.

**Фронтенд** (англ. **front-end**) — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

**Бэкенд (**[англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)**back-end) —** программно-аппаратная часть сервиса.

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

**Model-View-Controller (MVC, «Модель-Представление-Контроллер») —** схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три компонента: модель, представление и контроллер — таким образом, что изменение каждого компонента может осуществляться независимо.

* **Модель (Model)** предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* **Представление (View)** отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* **Контроллер (Controller)** интерпретирует действия пользователя, оповещая модель о необходимости изменений.

**ORM (Object-Relational Mapping) –** технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

**Document Object Model (DOM)** — это способ представления и взаимодействия с объектами в HTML, XHTML и XML документах. Согласно этой модели, каждый такой документ представляет собой иерархическое дерево элементов, называемое DOM-деревом.

**СУБД** **(Система управления базами данных)** - совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

**SQL** — декларативный язык программирования, применяемый для создания, изменения и управления данными в реляционной базе данных, управляемой соответствующей СУБД.

**API (англ. application programming interface)** — описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.

**Объектно-ориентированное программирование (ООП**) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

**Куки (англ. cookie)** — фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя.

**Прототипирование** — «черновая» реализация базового функционала будущего продукта/изделия, для анализа работы системы в целом.

# Обзор языков программирования

# 5.1. Python

Итак, серверная часть моего приложения разработана на языке **Python3**.

**Python 3** – высокоуровневый язык программирования. Ориентирован на повышение производительности разработчика и читаемость кода. Данный язык программирования часто используется для реализации серверной части веб-приложений, так как существуют различные Python-фреймворки, которые облегчают процесс написания на языке Python кода серверной части.

Почему именно Python? Рассмотрим преимущества и недостатки этого языка подробнее.

**Преимущества:**

**Универсальность** - Python не предназначен для каких-либо конкретных вещей, это язык общего назначения. Он подходит как для web-проектов, так и для desktop-  и серверных приложений.

**Мультипарадигменность** - Python поддерживает любой стиль программирования.

**Низкий порог вхождения** – этот язык программирования очень легок в изучении.

**Легко читаемый синтаксис** – позволяет легко читать чужой код и быстро разбираться в нем.

**Большое количество библиотек** - иногда для написания программы достаточно лишь найти подходящие модули и правильно их применить. Это позволяет думать о составлении программы на более высоком уровне, работая с уже готовыми элементами, которые выполняют различные действия.

**Переносимость** - Python реализован под всеми распространенными ОС и на множестве архитектур. Разворачивание приложений на другой машине происходит легко и быстро.

Автоматический сбор “мусора”. Отсутствие утечек памяти.

Потребляет довольно мало ресурсов, вследствие чего легче найти хостинги для Python.

**Недостатки:**

1. Динамическая типизация может приводить к огромному количеству ошибок в больших проектах.
2. Python – интерпретируемый язык программирования. Это снижает его быстродействие.
3. Из-за гибкости типов данных при работе с Python потребление памяти является значительным.

# 5.2. Java

**Java** — язык программирования общего назначения. Он следует парадигме ООП. Java используется для сетевых, мобильных, настольных и корпоративных приложений.

**Преимущества:**

1. Широкое распространение - Java это язык, который преподают в рамках введения в программирование в большинстве школ и университетов.
2. Этот язык считается безопасным так как не имеет указателей и в нем есть Security Manager – политика безопасности приложений, в которой можно настроить правила доступа.
3. Кроссплатформенность.
4. Java – язык для распределенного программирования. Он имеет встроенный механизм совместного использования данных и программ несколькими компьютерами, который повышает эффективность разработки.
5. Автоматическое управление памятью.
6. Многопоточность - Java позволяет запускать потоки одновременно, чтобы максимально эффективно использовать время процессора.

**Недостатки:**

1. Java занимает больше места в памяти. Это одна из проблем в языке Java, и разработчики пока не смогли преодолеть эту проблему.
2. Код на Java довольно многословен по сравнению с кодом на Python. Это усложняет разработку новичкам.

# 5.3. PHP

**PHP (Hypertext Preprocessor)–** это язык программирования, специально разработанный для написания web-приложений (сценариев), исполняющихся на web-сервере.

**Преимущества:**

1. Быстрое создание динамически генерируемых web-страниц.

2. Бесплатное распространение.

3. Кроссплатформенность - РНР перенесен на все основные ОС.

4. Простота в изучении.

**Недостатки:**

1. Непоследовательный синтаксис – когда язык был только создан, все ПО разрабатывалось с помощью С, поэтому много синтаксиса было заимствовано из него. Однако, современная аудитория больше ориентирована на Java. В итоге код переполнен различными языками.

2. Узкопрофильность – PHP предполагает программирование для интернета.

3. Низкая безопасность.

Таким образом, каждый язык программирования имеет свои преимущества и недостатки. Однако, ключевыми факторами при выборе языка разработки для меня являются читаемость кода и универсальность. Java уступает питону в первом пункте, а PHP во втором. Также мне хочется углубить свои навыки разработки на Python-е, полученные ранее.

# 6. Обзор фреймворков

# 6.1. Django

Фреймворк, который был использован мной в процессе разработки – **Django**.

**Django (Джанго)** — фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

Он существенно облегчает процесс разработки сайтов и веб-приложений. Данный фреймворк предоставляет большой спектр возможностей для разработки и имеет много дополнительных модулей, что позволяет снизить время разработки. Также Django предоставляет панель администрирования, модуль с аутентификацией и ORM, обеспечивая удобство разработки и поддержки приложений и позволяя ускорить процесс работы.

**Преимущества:**

1. **Быстрота** – Django является лучшим решением для разработчиков, для которых вопрос дедлайна стоит в приоритете, потому что он был разработан для того, чтобы помочь разработчикам создать приложение настолько быстро, насколько это возможно.

2. **Полная комплектация** – Django содержит огромное количество функциональности для решения большинства задач веб-разработки. Например, ORM, миграции базы данных, аутентификация пользователя, панель администратора, формы и многое другое.

3. **Безопасность** – Django включает механизмы предотвращения распространенных атак вроде SQL-инъекций (XSS) и подделки межсайтовых запросов (CSRF).

4. **Стандартизированная структура** – Django сам задаёт структуру проекта. Она помогает разработчикам понимать, где и как добавлять новый функционал.

5. **Django apps** – приложения в Django делят проект на несколько частей. Это позволяет легко интегрировать готовые решения.

6. **REST Framework** - Django REST Framework (DRF) является библиотекой для построения API. Он имеет модульную и настраиваемую архитектуру, которая хорошо работает для создания API.

**Недостатки:**

1. Django ORM значительно уступает последней SQLAlchemy.

2. Довольно медленное развитие - Django является большим фреймворком. С одной стороны, это позволяет сообществу разрабатывать множество универсальных модулей и приложений, но с другой - снижает скорость разработки самого Django.

3. Сложность - после каждого нового релиза увеличивается сложность работы с фреймворком.

# 6.2. Spring

**Spring Framework -** универсальный фреймворк с открытым исходным кодом для Java-платформы.

**Преимущества:**

1. Spring предоставляет каркас будущего приложения - есть определенная архитектура приложения, в которую разработчикам нужно встроить свою функциональность.
2. Внедрение зависимостей – в приложении на основе Spring объекты слабосвязаны и их необходимо строить на основе интерфейсов (зависимости внедряются в виде интерфейса для возможной последующей замены реализации).
3. Хорошая документация.
4. Гибкая конфигурация.
5. Spring избавляет нас от самостоятельного создания фабричных и синглтон-классов.

**Недостатки:**

1. Длительная процедура подготовки, связанная с его настройкой.
2. Кривая обучения - некоторым разработчикам трудно понять концепцию IOC и Dependency Injection. Из-за этого они не раскрывают весь потенциал Spring.
3. Параллельные механизмы. Одним из больших преимуществ Spring является то, что он предоставляет разработчикам широкий спектр возможностей, но это также может быть недостатком, поскольку вызывает путаницу. Разработчики должны знать, какие функции будут полезны, и принятие неправильных решений может привести к значительным задержкам.
4. Никаких особых рекомендаций. В документации Spring ничего не говорится о XSS угрозах или подделке межсайтовых запросов. Имея это в виду, вам и вашей команде нужно будет найти способы предотвратить проникновение хакеров в ваше приложение.

Подводя итог, для разработки моего веб-приложения был выбран фреймворк Django потому что для меня, как для новичка, важна хорошая документация и низкий порог вхождения.

# 6.3. Vue, React, Angular

Фронтенд — это одно из наиболее динамично развивающихся направлений современной разработки. Поэтому существует большое количество инструментов, библиотек и фреймворков, призванных помочь в работе.

Ниже я приведу краткое описание некоторых фреймворков для разработки пользовательских интерфейсов. Я не буду делать обзор преимуществ и недостатков данных фреймворков, потому что они не являются альтернативами Django. Однако они очень хороши в своей сфере и грамотному разработчику необходимо знать о них.

**Vue** — это JavaScript фреймворк для создания пользовательских интерфейсов. Он довольно быстро интегрируется в проекты с использованием других JavaScript-библиотек и может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

**Компании, использующие Vue:** Ozon, Спортмастер, GitLab, Nintendo, Grammarly и др.

**React** - JavaScript-библиотека для разработки пользовательских интерфейсов. Эта библиотека может использоваться для разработки одностраничных и мобильных приложений. Главные цели React - предоставить простоту, высокую скорость и масштабируемость.

**Компании, использующие React:** Yandex, Тинькофф, Мегафон, ВК, Netflix, WhatsApp, Facebook, Сбербанк и др.

**AngularJS** — JavaScript-фреймворк с открытым исходным кодом. Предназначен для разработки одностраничных приложений. Его цель — расширение браузерных приложений на основе MVC-шаблона и упрощение тестирования и разработки.

**Компании, использующие Angular:** Skyeng, Positive Technology, MTS, Google и др.

# 7. Обзор СУБД

# 7.1. SQLite

В моем проекте используется компактная встраиваемая СУБД - **SQLite3**.

**Преимущества:**

1. В Python есть встроенная поддержка SQLite, а в Django данная СУБД используется по умолчанию.

2. Файловая структура - вся база данных хранится в одном файле.

3. Полностью свободная лицензия.

4. Кроссплатформенность.

5. Отсутствие необходимости настройки сервера СУБД - у SQLite нет сервера, а все взаимодействие выполняет библиотека SQLite, без каких-либо дополнительных серверов.

6. Простота использования.

7. Высокая скорость на простых операциях.

8. С помощью SQLite можно очень быстро прототипировать.

**Недостатки:**

1. Нет многопользовательского доступа - открытие файла базы не предполагает параллельного доступа.
2. Отсутствие возможности увеличения производительности - исходя из проектирования SQLite нельзя сделать более производительной, поменяв настройки.
3. Не безопасна.
4. Отсутствие сервера может являться недостатком при разработке крупных проектов.

В качестве основных альтернатив к SQLite выступают MySQL и PostgreSQL.

# 7.2. MySQL

**MySQL** - свободная реляционная СУБД, разработку и поддержку которой осуществляет корпорация Oracle.

**Преимущества:**

1. Многофункциональность **-** MySQL обладает практически всем инструментарием, который может понадобиться в реализации проекта.
2. Безопасность - множество встроенных функций безопасности в этой СУБД работают по умолчанию.
3. Подходит для разработки крупных проектов.

**Недостатки:**

1. Довольно сложная установка вследствие необходимости настройки сервера.
2. Довольно сложно для начинающих.
3. Имеет функциональные ограничения - по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях.

# 7.3. PostgreSQL

**PostgreSQL** - свободная объектно-реляционная система управления базами данных.

**Преимущества:**

1. Дополнения - несмотря на огромное количество встроенных функций, PostrgreSQL имеет очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими.
2. Объектность - PostrgreSQL является не только реляционной СУБД, но и объектно-ориентированной с поддержкой наследования и много другого.

**Недостатки:**

1. Довольно трудно прототипировать.
2. Нужен хостинг.
3. Сложность работы с СУБД.

Таким образом, для работы над моим проектом мной была выбрана SQLite, потому что эта СУБД имеет встроенную поддержку в Python и используется по умолчанию в Django. Также, мое приложение – это учебный проект, для которого необходимы легкость работы с СУБД и возможность быстрого прототипирования. Конечно, если бы я разрабатывала очень крупный проект, то мною была бы выбрана одна из рассмотренных альтернатив.

# Front-end

Для написания клиентской части моего приложения использовались следующие языки программирования:

1. **HTML (HyperText Markup Language — «язык гипертекстовой разметки»)** — стандартизированный язык разметки документов в сети Интернет. Большинство веб-страниц размечены именно при помощи HTML (или XHTML).

2. **CSS (Cascading Style Sheets — каскадные таблицы стилей**) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Он преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML.

3. **JavaScript (JS)** - мультипарадигменный язык программирования, который поддерживает объектно-ориентированный, императивный и функциональный стили программирования. Является реализацией стандарта ECMAScript. Интерпретаторами для данного языка являются веб-браузеры.

# Техническое описание приложения

Итак, мной было решено разработать веб-приложение “MathHelp” для помощи студентам первого курса в освоении основ математического анализа.

MathHelp предоставляет возможность регистрации пользователя и получения им персонального аккаунта. На этой платформе студент может получить файл в формате pdf с теорией и задачами по каждому билету. Проверить свои знания можно с помощью тестов, которые также разделены по темам. Каждый билет имеет специальный блок, где пользователь может оставить комментарий о билете или свое решение задачи.

# 9.1. Структура проекта на Django

Django использует термин “проект” для всего разрабатываемого веб-приложения. В этом фреймворке используется модульная система приложений. Это означает, что одно приложение состоит из нескольких приложений, отвечающих каждое за свой функционал. Такой подход позволяет использовать уже написанные модули в разных проектах и выстроить логическую структуру проекта в целом.

В каждом приложении существуют следующие файлы:

1. **urls.py**

* содержат роуты (routes)
* задают соответствие между url-ами и обработчиками запросов (view)

1. **views.py**

* содержит view (на вход view получает объект-запрос, на выход отдают объект-ответ)
* чаще всего объект-ответ включает в себя сгенерированный с помощью шаблонов html

1. **templates (шаблоны)**

* директория для хранения файлов с кодом html (в коде используются также инструкции на языке Django-шаблонизатора)
* позволяют легко генерировать html по входным данным

1. **models.py**

* содержат модели данных (models) – это специальные классы, описывающие элементы, хранимые в базе данных
* на их основе ORM генерирует методы работы с базой данных - через них можно искать, создавать, изменять и удалять объекты в базе

1. **forms.py**

* содержат формы
* по декларативному описанию умеют создавать html-формы, а также обрабатывать данные от них на стороне сервера

# 9.2. Структура приложений

Мой проект имеет несколько приложений, каждое из которых отвечает за свой функционал.

Для наглядности я приведу таблицу моих приложений, каждое из которых описано по схеме MVC:

|  |  |  |  |
| --- | --- | --- | --- |
| **приложение** | **models.py** | **templates** | **views.py** |
| **Tickets** | Ticket | ticket-detail.html  ticket-list.html | TicketDetailView  TicketListView  TicketPDFView |
| **Forum** | Comment | - | CommentCreateView  CommentDeleteView |
| **Quiz** | Quiz  Question  Answer  Result | quiz-form.html  quiz-report.html  quiz-result.html | QuizFormView  QuizSaveView  QuizRestartView  QuizReportView  QuizResultView |
| **Profiles** | Profile | profile-detail.html  profile-update.html | ProfileDetailView  ProfileUpdateView  ProfileRedirectView |
| **Pages** | - | index.html  references.html  footer.html  anon-header.html  auth-header.html | IndexView  ReferencesView |
| **Accounts** | - | account-confirm-delete.html  account-delete.html  account-email-update.html  account-login.html  account-password-change.html  account-username-update.html  account-settings.html | AccountDeleteView  AccountEmailUpdateView  AccountLoginView  AccountPasswordChangeView  AccountUsernameUpdateView  AccountCreateView  AccountSettingsView |

**Примечание:** приложение tests не описано в таблице, потому что оно является вспомогательным для разработчика, и пользователь не имеет к нему доступа.

# 9.3. Django Models

Веб-приложения Django получают доступ и управляют данными через объекты Python, называемые **моделями**.

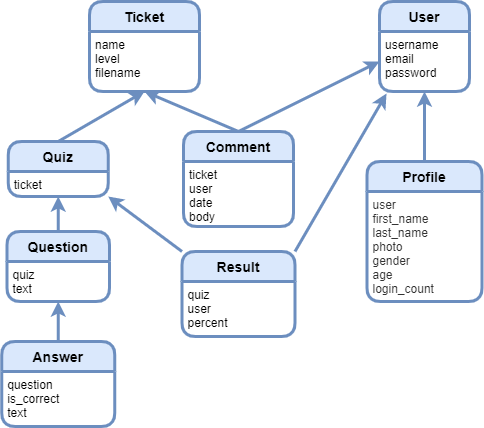
Модели определяют структуру хранимых данных, включая типы полей и, возможно, их максимальный размер, значения по умолчанию, параметры списка выбора, текст справки для документации, текст меток для форм и т. д.

- Каждая модель представляет собой класс Python, который является подклассом django.db.models.Model.

- Каждый атрибут модели представляет поле базы данных.

- При этом Django предоставляет вам автоматически сгенерированный API доступа к базе данных.

Для удобства хранения и работы с данными в моем проекте описано несколько таких моделей, структура которых представлена ниже.

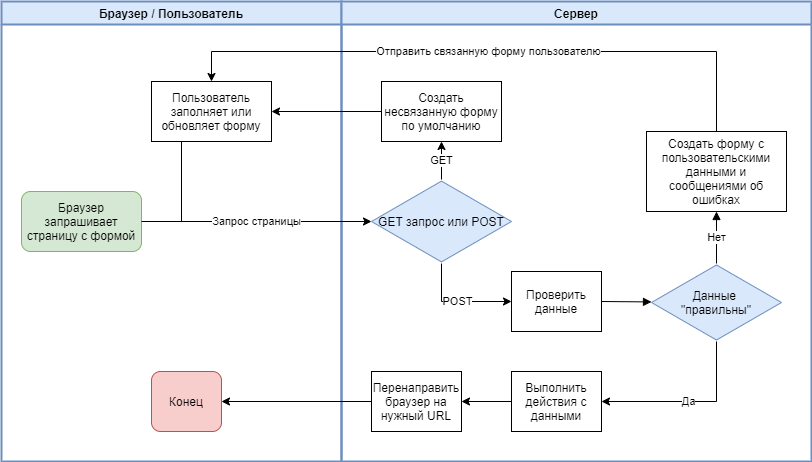


Структура базы данных

# 9.4. Работа с формами

Все данные, которые я получаю от пользователей, проходят через формы, которые описаны в файле forms.py.

Диаграмма, демонстрирующая процесс работы с формами в Django:



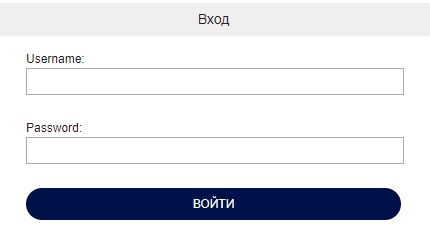
Процесс работы с формами в Django

HTTP определяет несколько методов запроса, которые указывают, какое желаемое действие выполнится для данного ресурса. На диаграмме показаны только методы GET и POST, однако в моем приложении используются также PUT и DELETE.

* **GET** - запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.
* **POST** - используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.
* **PUT** - заменяет все текущие представления ресурса данными запроса.
* **DELETE** - удаляет указанный ресурс.

**Этапы работы с формой на примере входа в аккаунт на MathHelp:**

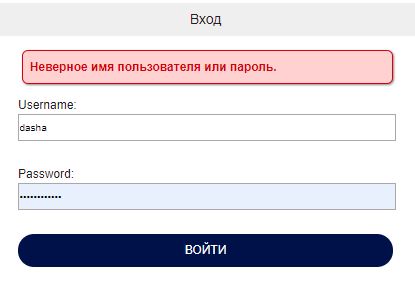
1. Показ дефолтной формы при первом запросе от пользователя:



1. Получение данных из формы со стороны клиента и связывание их с формой на стороне сервера.
2. Очистка и валидация данных.

* Очистка данных - удаление неправильных символов, которые потенциально могут использоваться для отправки вредоносного содержимого и конвертация очищенных данных в подходящие типы данных Python.
* Валидация – проверка соответствия введенных данных типу поля модели.

1. Если данные неверны, то происходит перерисовка формы с уже введенными пользователем данными и сообщением об ошибке:



Сообщения об ошибках описаны в классе соответствующей формы (AccountLoginForm) в виде словаря:

**error\_messages = {**

**'invalid\_login': 'Неверное имя пользователя или пароль.',**

**}**

1. Если все данные верны, то пользователь будет перенаправлен на страницу его профиля.

# 9.5. Сессии

Мое приложение использует протокол HTTP при взаимодействиях между браузерами и серверами. Однако, этот протокол не сохраняет свое состояние. Т.е. когда пользователь открывает одну страничку, а затем другую, HTTP не может установить, что оба запроса принадлежат одному пользователю. Но на практике существуют задачи, для которых необходимо отследить действия пользователя. Для их решения существует особый метод отслеживания – управление сеансами (сессиями).

**Сессия –** это вспомогательный логический объект, который способствует передаче данных между последовательными HTTP – запросами от одного пользователя.

Django использует cookie, которые содержат специальный идентификатор сессии. Реальные данные сессии, по умолчанию, хранятся в базе данных сайта, что более безопасно.

Мы можем получить доступ к переменной session в соответствующем отображении через параметр request (HttpRequest передается как первый аргумент в каждое отображение). Переменная сессии – это связь с определенным браузером (пользователем), который определяется при помощи идентификатора сессии, получаемого из cookie браузера.

Переменная session – это объект-словарь, который служит для чтения и записи неограниченное число раз.

Ниже представлен фрагмент кода, который показывает, как получать, задавать и удалять некоторые данные при помощи ключа 'read\_ticket', связанного с текущей сессией (браузером).

# получение значения сессии при помощи ключа ('read\_ticket')

# если такого ключа нет, то возникнет ошибка KeyError

**read\_ticket = request.session['read\_ticket']**

# передача значения в сессию

**request.session['read\_ticket'] = 12**

# удаление значения из сессии

**del request.session['read\_ticket']**

В моем веб-приложении сессии используются для получения следующей информации о действиях пользователя:

* прочитанные билеты
* пройденные тесты
* текущие результаты по тестам

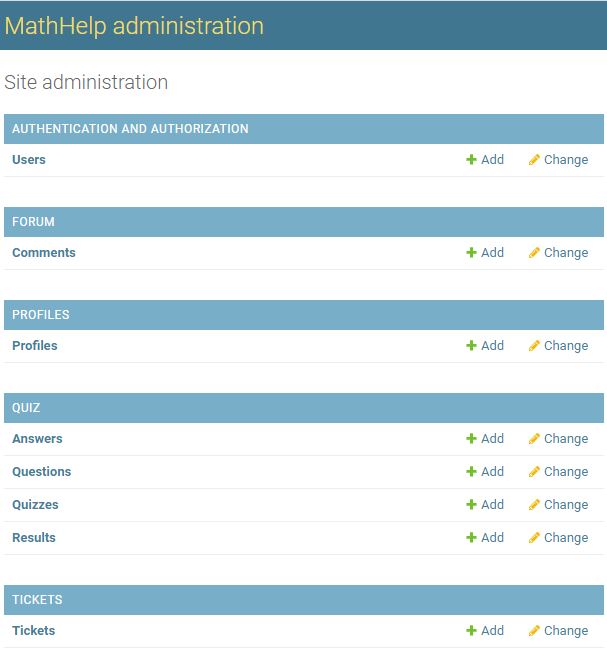
# 9.6. Возможности пользователя панели администратора

Приложение Django admin на основе ваших моделей может автоматически создать часть сайта, предназначенную для создания, просмотра, обновления и удаления записей. Эта часть сайта называется панелью администратора.

Для того, чтобы добавить модели в приложение admin, необходимо их зарегистрировать в файле admin.py:

**@admin.register(Question)**

Для того, чтобы войти в панель администратора необходимо иметь учетную запись пользователя со статусом Staff или SuperUser. Я использовала последнюю, потому что она дает полный доступ к сайту и все необходимые разрешения.

Ниже представлен скриншот моей панели администратора.

Панель администратора MathHelp

Я немного изменила конфигурацию отображения моих моделей для того, чтобы было проще с ними работать:

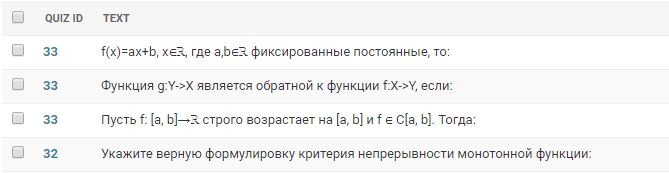
1. добавила фильтры списка с помощью list\_filter
2. и настроила отображение полей с помощью list\_display.

**class QuestionAdmin(admin.ModelAdmin):**

**list\_display = ['quiz\_id', 'text']**

**list\_filter = ['quiz\_id']**

Cкриншот отображаемых полей модели Question:



# Заключение

Итак, в процессе разработки серверной части приложения я улучшила свой навык написания кода на языке Python, начала осваивать новое для меня средство разработки – фреймворк Django и получила навык работы с СУБД SQLite. В частности, я ознакомилась со структурой проекта на Django и с синтаксисом данного фреймворка. Мне было интересно узнать о его возможностях и поработать с MVC архитектурой. Я также получила такие ценные навыки как: создание модели базы данных, работа с сессиями и формами.

Мне также удалось углубить свои знания в областях HTML, CSS и JavaScript во время разработки frontend-части моего веб-приложения.

Во время работы над теоретической частью я ознакомилась с преимуществами и недостатками рассмотренных мной средств разработки. Это, безусловно, полезный опыт, потому что каждый программист должен быть осведомлен о современных средствах разработки и их особенностях.

Также, мною были получены знания о работе web-сайтов и web-серверов, что было одой из целей моей работы.

И, наконец, я разработала собственное веб-приложение для помощи студентам в изучении основ математического анализа, которое вы можете найти по адресу: **http://dselivanovskaya.pythonanywhere.com/**

# Список литературы и ссылок

Django:

1. <https://djbook.ru/rel1.9/>
2. <https://docs.djangoproject.com/en/3.0/>
3. <https://developer.mozilla.org/ru/docs/Learn/Server-side/Django>
4. <https://ru.wikipedia.org/wiki/Django>
5. https://tutorial.djangogirls.org/ru/
6. “Django. Разработка веб-приложений на Python” Форсье Джефф, Биссекс Пол, Чан Уэсли Дж

HTML, CSS:

1. <http://htmlbook.ru/>
2. <http://htmlbook.ru/samcss>
3. <https://ru.wikipedia.org/wiki/HTML>
4. <https://ru.wikipedia.org/wiki/CSS>
5. <https://openedu.ru/course/ITMOUniversity/WEBDEV/>

Сессии:

1. <https://developer.mozilla.org/ru/docs/Web/HTTP/Session>

Python:

1. <https://pythonworld.ru/samouchitel-python>
2. <https://ru.wikipedia.org/wiki/Python>

СУБД:

1. <http://www.mysql.ru/docs/man/What-is.html>
2. <https://www.opennet.ru/docs/RUS/what_is_pgsql/>
3. <https://ru.wikipedia.org/wiki/SQLite>
4. <https://ru.wikipedia.org/wiki/PostgreSQL>
5. <https://ru.wikipedia.org/wiki/MySQL>

Научные статьи:

1. “Сравнительный анализ javascript-фреймворков.”

Никулушкина Ю.А., Шуклин Д.А.

<https://f5f7e10c-a5dd-471b-96cb-d1595326738e.filesusr.com/ugd/b06fdc_0e7a2fce794a4d4f871ba31a1c4f1c6b.pdf?index=true> [Страницы: 645-648]

1. “Сравнение функциональных возможностей и структуры фреймворков Django и ASP.NET MVC, реализующих паттерн программирования MVC.”

Афанасьев Г.И., Григорьев Е.А.

<https://f5f7e10c-a5dd-471b-96cb-d1595326738e.filesusr.com/ugd/b06fdc_55348c3bae7a4eeea5cc7be5e8914b1f.pdf?index=true>[Страницы: 914-926]

1. “Обзор современных фреймворков и инструментов, используемых для разработки web-приложений”

Байнов А. М., Кривоногова А. Е., Николаев А. С., Богомолова О. И.

Журнал “Наука без границ”

<https://nauka-bez-granic.ru/%e2%84%96-1-41-2020/>[Страницы: 19-23]

Другое:

1. <http://proglang.su/java/introduction-to-programming>
2. <https://ru.wikipedia.org/wiki/Java>
3. <https://ru.wikipedia.org/wiki/PHP>
4. <http://www.mirsite.ru/php.htm>
5. <https://ru.wikipedia.org/wiki/Vue.js>
6. <https://ru.wikipedia.org/wiki/React>
7. <https://ru.wikipedia.org/wiki/AngularJS>