Министерство образования и науки Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук
Кафедра программной инженерии

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
д-р. техн. наук, профессор
_____ С.П. Сущенко
«08» 06 2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА САЙТА НАУЧНЫХ КОНФЕРЕНЦИЙ

по основной образовательной программе подготовки бакалавров
направление подготовки 02.03.03 – Математическое обеспечение и администрирование информационных систем

Лауэр Марк Евгеньевич

Руководитель ВКР
д-р. физ.-мат. наук, доцент
_____ А.Н. Моисеев
«31» Мая 2018 г.

Автор работы
студент группы №1443
_____ М.Е. Лауэр

Томск-2018

РЕФЕРАТ

Выпускная квалификационная работа 30 с., 26 рис., 12 источников.

WEB, FLASK, REACT, РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ.

Цель работы – разработать систему управления содержимым для сайтов научных конференций.

Результаты работы – выявлены требования к системе, составлена модель предметной области, спроектирована архитектура системы, реализован проект.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Система представляет собой веб-приложение. Основная ее задача – отображение и работа с информацией о научных конференциях. Пользователь может просматривать конференции и зарегистрироваться в качестве ее участника. Будучи автором или докладчиком научной статьи, он может опубликовать свою статью на сайте. Работа с конференциями осуществляется администратором приложения.

Цель работы: разработать систему управления содержимым для сайтов научных конференций.

Задачи:

- выявить требования к системе;
- составить модель предметной области;
- спроектировать архитектуру системы;
- реализовать проект.

1 Определение и анализ требований

1.1 Требования

В ходе обсуждения проекта с заказчиком были выявлены функциональные и нефункциональные требования. В соответствии с функциональными система должна:

- предоставлять пользователю возможность просмотра следующей информации о конференциях:

1) название конференции;

2) место проведения;

3) даты проведения;

4) цели;

5) тематика;

6) программа;

7) рабочие языки;

8) важные даты;

9) порядок публикации статей;

10) организационные взносы;

11) международный программный комитет;

12) организационный комитет;

13) контакты организационного комитета;

14) партнеры;

15) организаторы;

16) спонсоры;

17) совместные мероприятия;

18) прочая информация;

- предоставлять администратору инструменты для создания, редактирования, изменения и удаления конференций;

- предоставлять администратору инструменты для изменения дизайна сайта;

- предоставлять пользователю возможность регистрации в качестве слушателя, автора или докладчика;

- предоставлять пользователю возможность отправить свою научную работу и следить за ее статусом, если пользователь является автором или докладчиком;

- предоставлять администратору возможность формирования информационного письма на основе конференции и его массовой рассылки;

- предоставлять администратору инструменты для создания, редактирования, изменения и удаления отдельных веб-страниц.

Нефункциональное требования:

- система должна представлять собой веб-приложение;

- участником конференции может быть только зарегистрированный пользователь.

1.2 Варианты использования

На рисунке 1 представлена диаграмма вариантов использования, выявленных из требований.

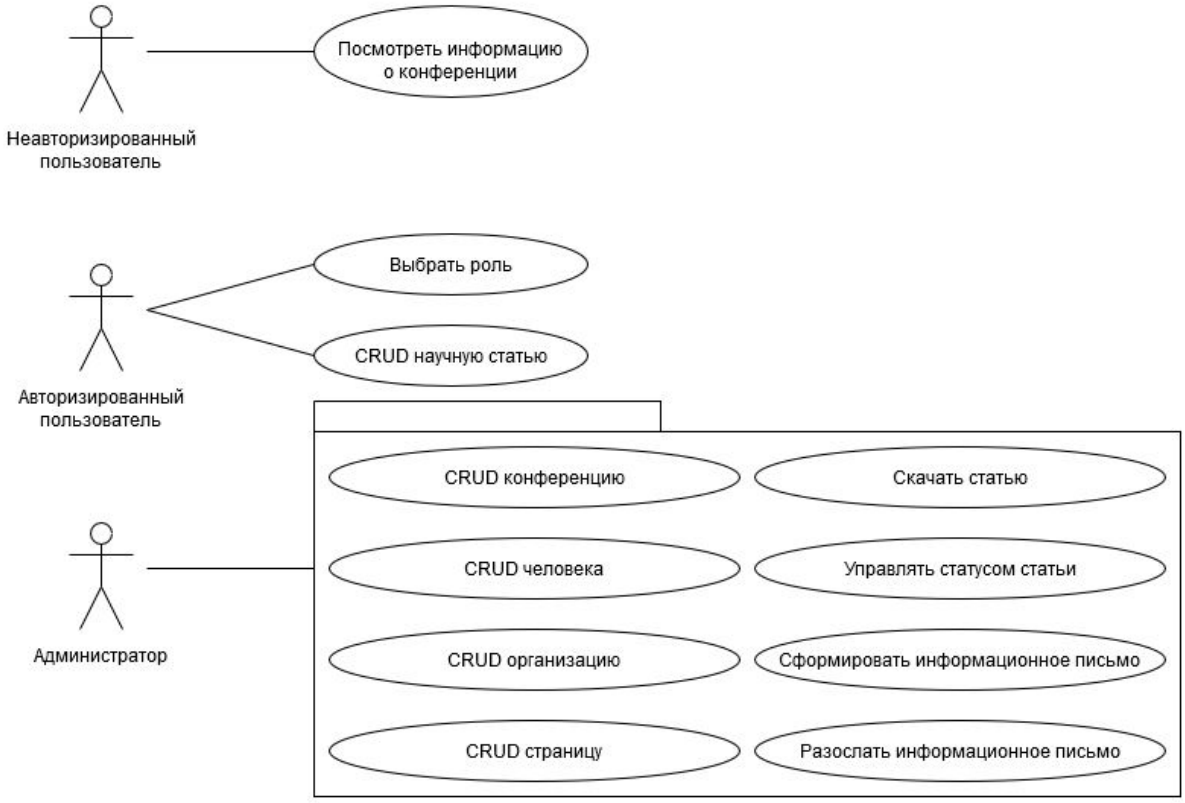


Рисунок 1 – Диаграмма вариантов использования

Из них архитектурно значимыми являются: CRUD конференцию, CRUD человека, CRUD организацию. Так как сценарии этих вариантов использования очень схожи между собой, в таблице 1 представлено описание только одного из них.

Таблица 1 – Сценарий варианта использования «Создать конференцию»

Название	Создать конференцию
Цель	Создать конференцию
Актер	Администратор
Предусловия	Пользователь находится в разделе «Управление конференциями»
Сценарий	1. Пользователь нажимает на кнопку пользовательского интерфейса «Создать конференцию». 2. Система проверяет, обладает ли пользователь правами на создание конференций.

Продолжение таблицы 1

	3. Система отображает форму для создания конференции. 4. Пользователь заполняет форму и нажимает на кнопку «Создать». 5. Система проверяет данные, введенные пользователем. 6. Система сохраняет конференцию в базе данных. 7. Система сообщает пользователю об успешном завершении создания. 8. Система переадресовывает пользователя в раздел «Управление конференциями».
Расширения	2.1. Пользователь не обладает необходимыми правами. 2.1.1. Система выводит пользователю сообщение об ошибке. 5.1. Данные не прошли проверку. 5.1.1. Система выводит пользователю сообщение об ошибке. 5.1.2. Пользователь исправляет введенные данные.

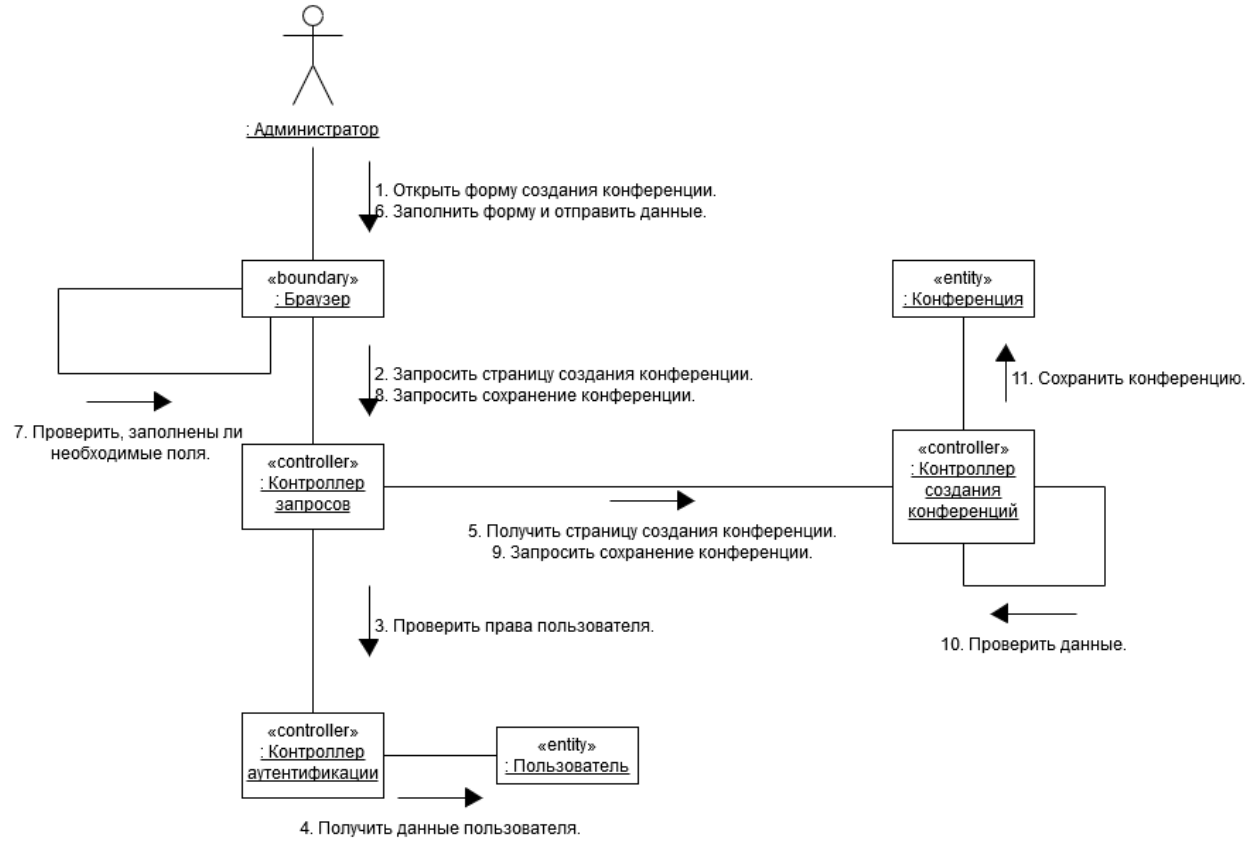


Рисунок 2 – Диаграмма анализа варианта использования «Создать конференцию»

Другие варианты использования для создания объектов в системе исполняются по схожему сценарию: по запросу пользователя система сначала проверяет его права, в случае успешной проверки отображает форму для заполнения, затем производится проверка в браузере (например, введены ли данные в обязательных полях) и в контроллере (например, существует ли в базе сущность со значением уникального атрибута, равным тому, что было введено пользователем), и только при соблюдении всех условий проверки добавляет данные в базу, в противном же случае выдает пользователю сообщение об ошибке.

Ниже описаны сценарии вариантов использования «Сформировать информационное письмо» и «Разослать информационное письмо».

Таблица 2 – Сценарий варианта использования «Сформировать информационное письмо»

Название	Сформировать информационное письмо
Цель	Сформировать информационное письмо для дальнейшей рассылки
Актер	Администратор
Предусловия	Пользователь находится на странице конференции
Сценарий	1. Пользователь нажимает на кнопку «Сформировать письмо» 2. Система проверяет, обладает ли пользователь соответствующими правами. 3. Система отображает страницу формирования письма с полями выбора данных конференции и полем для ввода дополнительной информации. 4. Пользователь заполняет форму и нажимает на кнопку «Сформировать». 5. Система проверяет введенные пользователем данные. 6. Система формирует письмо. 7. Система переадресовывает пользователя на страницу сформированного письма.
Расширения	2.1. Пользователь не обладает необходимыми правами. 2.1.1. Система выводит пользователю сообщение об ошибке. 5.1. Данные не прошли проверку. 5.1.1. Система выводит пользователю сообщение об ошибке. 5.1.2. Пользователь исправляет введенные данные.

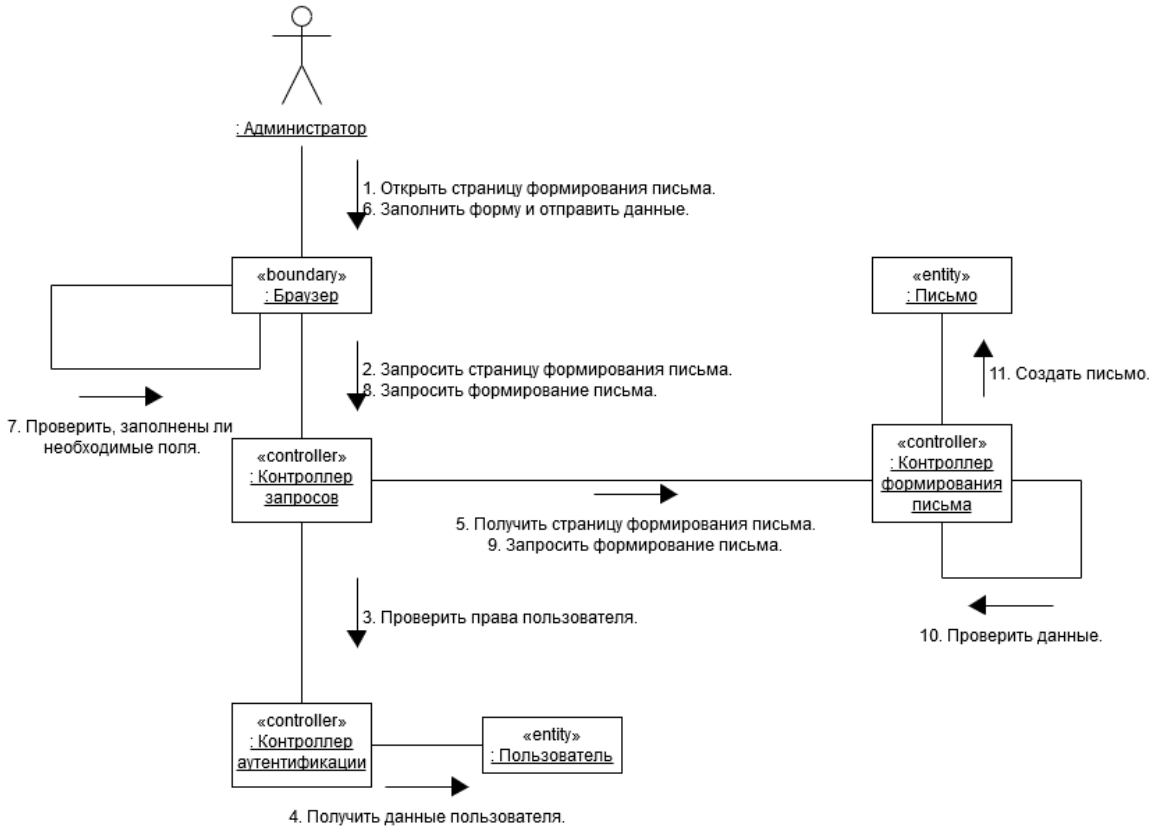


Рисунок 3 – Диаграмма анализа варианта использования «Сформировать информационное письмо»

Информационное письмо формируется на основе данных конкретной конференции. Администратор может выбрать какие данные будут в письме, а также может добавить к письму дополнительную информацию.

Таблица 3 – Сценарий варианта использования «Разослать информационное письмо»

Название	Разослать информационное письмо
Цель	Разослать информационное письмо по списку электронных адресов
Актер	Администратор
Предусловия	Пользователь находится на странице сформированного письма
Сценарий	1. Пользователь нажимает на кнопку «Разослать письмо». 2. Система проверяет, обладает ли пользователь соответствующими правами. 3. Система отображает список адресатов и поле для ввода дополнительных адресов. 4. Пользователь выбирает адресаты и нажимает на кнопку «Разослать». 5. Система проверяет введенные пользователем данные. 6. Система рассылает письмо по указанным адресам. 7. Система сообщает пользователю об успешном завершении рассылки. 8. Система переадресовывает пользователя на страницу письма.
Расширения	2.1. Пользователь не обладает необходимыми правами. 2.1.1. Система выводит пользователю сообщение об ошибке. 5.1. Данные не прошли проверку. 5.1.1. Система выводит пользователю сообщение об ошибке. 5.1.2. Пользователь исправляет введенные данные.

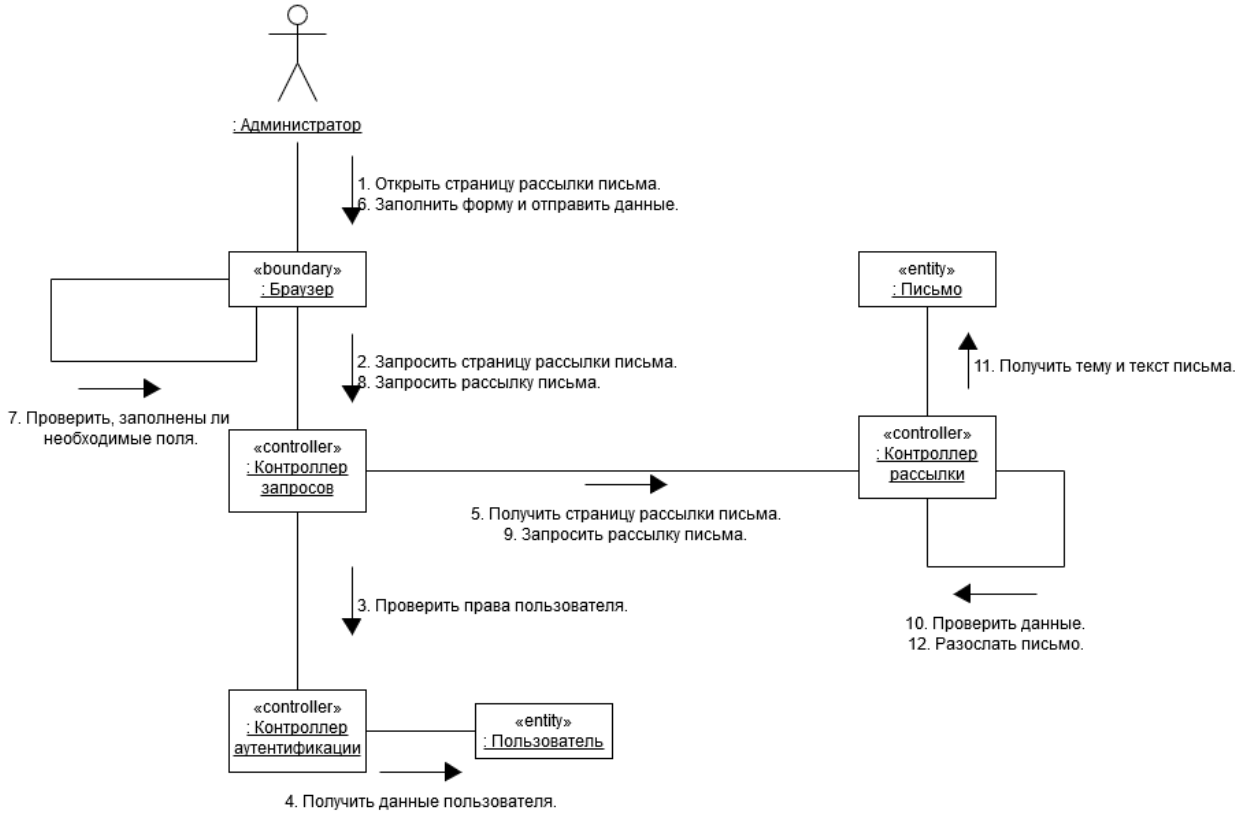


Рисунок 4 – Диаграмма анализа варианта использования «Разослать информационное письмо»

В таблице 4 представлено описание сценария варианта использования «Скачать статью».

Таблица 4 – Сценарий варианта использования «Скачать статью»

Название	Скачать статью
Цель	Скачать научную статью одного из участников конференции
Актер	Администратор
Предусловия	В системе создана хотя бы одна статья
Сценарий	1. Пользователь переходит на страницу участника. 2. Система отображает страницу выбранного участника. 3. Пользователь нажимает на кнопку «Научные статьи». 4. Система проверяет, обладает ли пользователь соответствующими правами. 5. Система отображает пользователю список научных статей участника. 6. Пользователь выбирает статью и нажимает на кнопку «Скачать». 7. Система переадресовывает пользователя на адрес файла статьи. 8. Система переадресовывает пользователя на страницу со списком статей участника.
Расширения	2.1. Участник не добавил ни одной статьи. 2.1.1. Система не отображает пользователю кнопку «Научные статьи». 4.1. Пользователь не обладает необходимыми правами. 4.1.1. Система выводит пользователю сообщение об ошибке.
Альтернативный сценарий 1	1. Пользователь переходит на страницу со списком научных статей всех участников. 2. Система проверяет, обладает ли пользователь соответствующими правами. 3. Пользователь выбирает статью и нажимает на кнопку «Скачать». 4. Система переадресовывает пользователя на адрес документа со статьей. 5. Система переадресовывает пользователя на страницу со списком статей всех участников.
Расширения альтернативного сценария 1	2.1. Пользователь не обладает необходимыми правами. 2.1.1. Система выводит пользователю сообщение об ошибке.

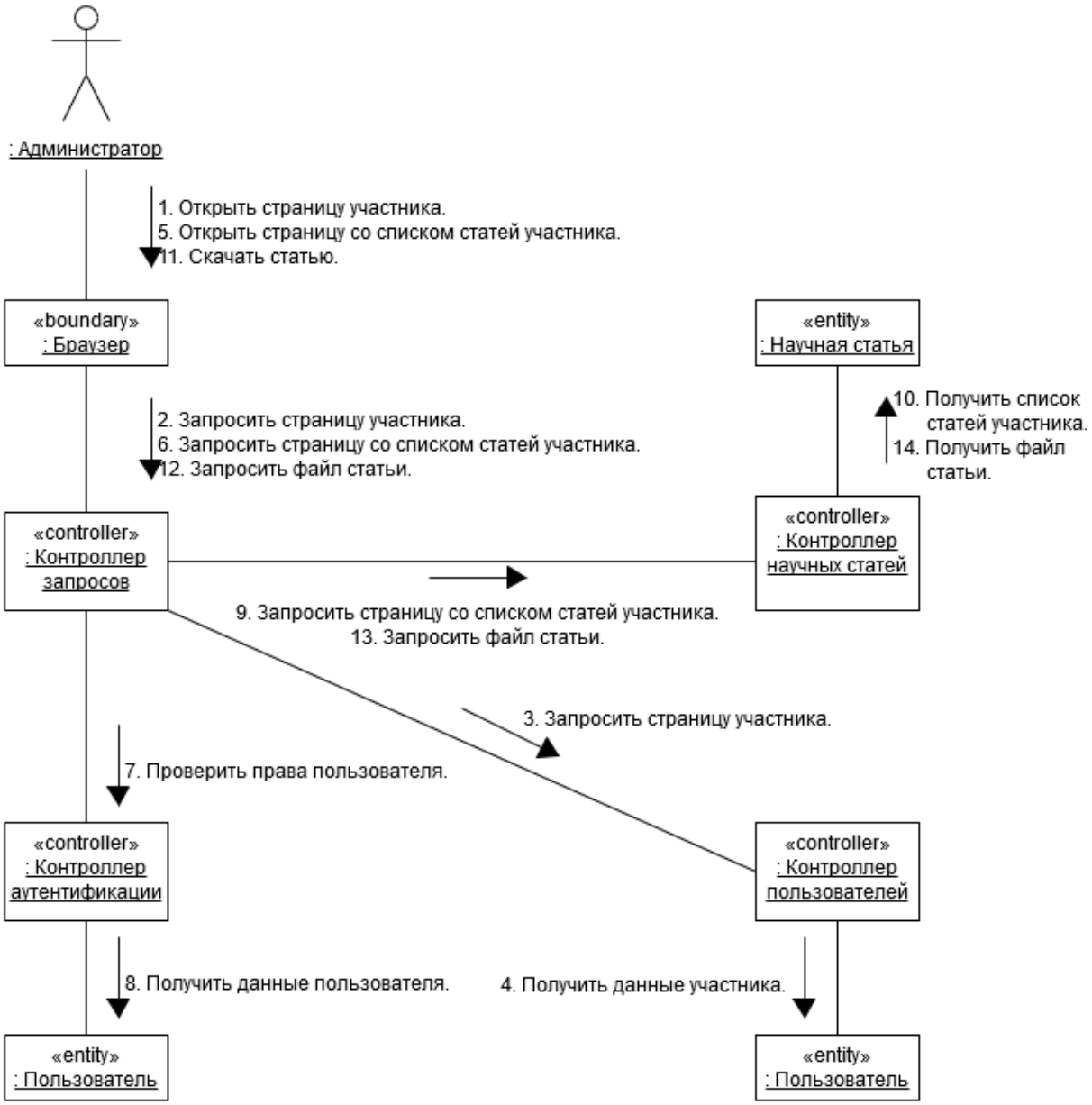


Рисунок 5 – Диаграмма анализа варианта использования «Скачать статью»

Сценарий варианта использования «Управлять статусом статьи» схож со сценарием варианта использования «Скачать статью» за исключением того, что администратор вместо нажатия на кнопку «Скачать» выбирает статус статьи из предложенных вариантов.

Зарегистрированный пользователь может выбрать одну из ролей: слушатель, автор или докладчик. В таблице 5 представлено описание сценария этого варианта использования.

Таблица 5 – Сценарий варианта использования «Выбрать роль»

Название	Выбрать роль
Цель	Выбрать роль
Актер	Авторизированный пользователь
Предусловия	Пользователь находится на странице своего профиля

Продолжение таблицы 5

Сценарий	1. Пользователь нажимает на кнопку «Редактировать». 2. Система проверяет, обладает ли пользователь соответствующими правами. 3. Система отображает страницу с формой редактирования профиля пользователя. 4. Пользователь выбирает роль из списка в соответствующем поле и нажать на кнопку «Сохранить». 5. Система проверяет введенные пользователем данные. 6. Система рассылает письмо по указанным адресам. 7. Система сообщает пользователю об успешном завершении рассылки. 8. Система переадресовывает пользователя на страницу письма.
Расширения	2.1. Пользователь не обладает необходимыми правами. 2.1.1. Система выводит пользователю сообщение об ошибке. 5.1. Данные не прошли проверку. 5.1.1. Система выводит пользователю сообщение об ошибке. 5.1.2. Пользователь исправляет введенные данные.

Рисунок 6 – Диаграмма анализа варианта использования «Выбрать роль»

Сценарий варианта использования «Посмотреть информацию о конференции»:

- пользователь открывает страницу конференции;
- система отображает страницу конференции со всей информацией о ней.

1.3 Модель предметной области

На рисунке 7 представлена модель предметной области в виде диаграммы классов.

Рисунок 7 – Диаграмма классов модели предметной области

Пояснения к модели предметной области:

- МПК – международный программный комитет;
- ОК – организационный комитет;
- совместные мероприятия – конференции, которые проводятся совместно, например, в одном месте и в те же даты.

2 Проектирование

2.1 Архитектура приложения

В качестве основы для разрабатываемого веб-приложения был выбран шаблон проектирования Модель-Представление-Контроллер (Model-View-Controller, MVC) [1]. Такой выбор обусловлен прежде всего тем, что такая архитектура позволяет разрабатывать отдельные компоненты приложения независимо от остальных, а также тем, что большинство веб-приложений основаны на этом шаблоне проектирования либо на его модификациях, соответственно большая часть инструментов разработки веб-приложений ориентирована именно на этот шаблон.

Архитектура приложения представлена на рисунке 8.

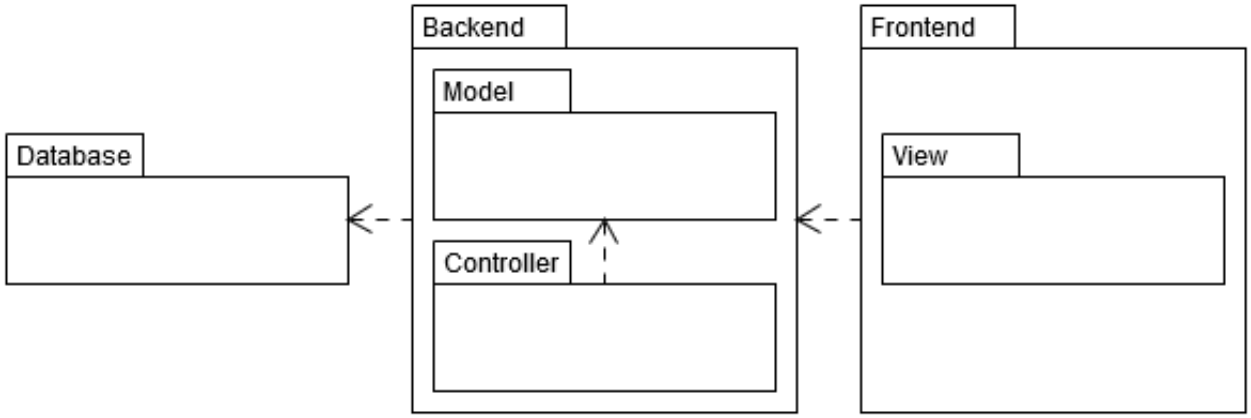


Рисунок 8 – Диаграмма пакетов архитектуры приложения

Здесь серверная часть (backend) представляет собой Web API, то есть на запросы клиентской части (frontend) серверная часть возвращает не HTML-документы, а данные в заданном формате, которые затем будут обработаны на клиентской части, то есть на их основе будет построено представление. Такой подход также позволяет реализовать одностраничное приложение – веб-приложение, использующее единственный HTML-документ как оболочку для всех веб-страниц и организующее взаимодействие с пользователем через динамически подгружаемые HTML, CSS и JavaScript файлы [2].

Алгоритм работы такого веб-приложения:

1) пользователь переходит по адресу веб-приложения;

2) серверная часть приложения возвращает пользователю один HTML-документ и сопутствующие файлы (JavaScript и CSS). Дальнейшее взаимодействие пользователя с приложением происходит через клиентскую его часть, исполняющуюся в браузере. Клиентская часть приложения представляет собой JavaScript файл (либо несколько файлов), который был отправлен при первом запросе пользователя;

3) пользователь взаимодействует с приложением (нажимает на кнопки пользовательского интерфейса, переходит по ссылкам и т.п.);

4) клиентская часть приложения обрабатывает все запросы пользователя и при необходимости получения каких-либо данных выполняет асинхронный запрос к серверной части;

5) серверная часть обрабатывает запрос: соответствующий контроллер принимает запрос, обрабатывает пришедшие с запросом дополнительные данные (если они есть), при необходимости обращается к модели за нужными данными, а затем формирует ответ;

6) серверная часть отправляет ответ клиенту;

7) клиентская часть принимает ответ и на основе полученных данных формирует представление;

8) переход на шаг 3.

Диаграмма деятельности для этого алгоритма представлена на рисунке 9.

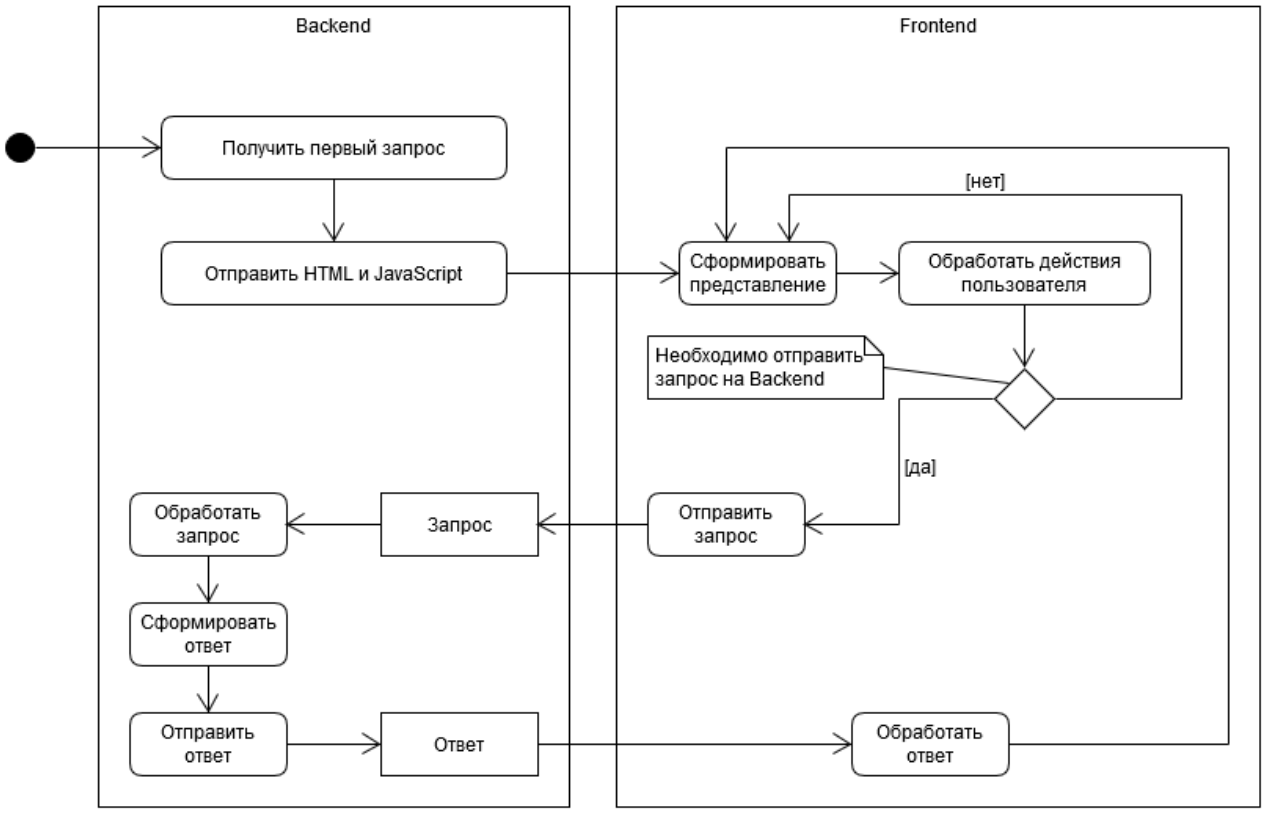


Рисунок 9 – Диаграмма деятельности работы приложения

2.2. Инструменты и технологии

2.2.1 Серверная часть

Для разработки серверной части приложения был выбран фреймворк Flask [3]. Этот фреймворк написан на языке программирования Python и является так называемым микрофреймворком. Микрофреймворк – это минималистичный каркас приложений, предоставляющий лишь базовые возможности.

Компоненты фреймворка Flask:

- Werkzeug – Python-библиотека для WSGI-приложений. WSGI (Web Server Gateway Interface) – стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером [4]. Эта библиотека предоставляет набор функций для работы с HTTP-запросами и ответами;

- Jinja – система веб-шаблонов. Позволяет создавать шаблоны веб-страниц на основе данных, полученных от контроллеров. При разработке приложения, описываемого в данной работе, использоваться не будет, так как представление будет формироваться клиентской частью приложения.

Выбор этого фреймворка обусловлен тем, что не требуется формировать представление на сервере, архитектурно значимые варианты использования подразумевают, что основная задача серверной части – выполнять запросы к базе данных и формировать ответ на основе результатов этих запросов.

Для разработки серверной части также были выбраны следующие инструменты:

- flask-login – библиотека для работы с авторизацией пользователей приложения [5];

- SQLAlchemy – библиотека для работы с реляционными СУБД с применением технологии ORM [6];

- flask-migrate – библиотека для работы с миграциями SQLAlchemy [7], миграция – это обновление схемы базы данных в соответствии с моделью приложения;

- marshmallow – библиотека для преобразования объектов SQLAlchemy в объекты базовых типов Python [8];

- sqlite – реляционная СУБД, является встраиваемой, то есть не запускается отдельным процессом, а представляет собой библиотеку, с которой компонуется программа [9]. Сама база данных хранится в единственном файле на компьютере, на котором программа исполняется.

Приложение, созданное с помощью Flask, представляет собой программу на языке Python, запускаемую на сервере. Контроллер в такой программе – это класс, включающий в себя методы, вызывающиеся при запросе по определенному URL, к которому этот метод

привязан. Например, сервер доступен по адресу http://1.2.3.4:8080/, а в контроллере есть метод, привязанный к пути “/home”. Тогда при обращении по адресу http://1.2.3.4:8080/home будет вызван этот метод.

Модель представляет собой набор классов, на основе полей которых будет построена схема базы данных. Также эти классы включают в себя методы для работы с полями.

2.2.2 Клиентская часть

Для разработки клиентской части приложения была выбрана библиотека React, написанная на языке программирования JavaScript [10]. Основная задача этой библиотеки – синхронизация пользовательского интерфейса с внутренним состоянием приложения.

Основные особенности этой библиотеки:

- компоненты – функции/классы для создания HTML-элементов и управления ими;
- связывание данных через свойства (props) – свойства передаются дочернему компоненту от родительского в виде JavaScript объекта, дочерний компонент может использовать эти свойства для манипуляций над своими HTML-элементами;
- связывание данных через состояние (state) – компонент может хранить некоторое состояние (в виде JavaScript объекта), которое также может быть использовано для управления HTML-элементами компонента;
- Virtual DOM – легковесная копия «настоящего» DOM (Document Object Model – объектная модель документа), это структура данных, которую создает React, при создании или изменении компонентов сначала заполняется эта структура, а затем React производит «сверку» данных в Virtual DOM с данными в реальном DOM и при необходимости меняет HTML-элементы в реальном DOM, такой подход позволяет сильно ускорить работу клиентской части приложения.

React-компоненты обладают жизненным циклом – набором методов, вызываемых на разных стадиях «жизни» компонента. На рисунке 10 представлена диаграмма деятельности жизненного цикла компонента.

Переопределяя эти методы, можно изменить стандартное поведение компонентов. Жизненный цикл компонента разделен на три стадии:

- Mounting (монтирование) – компонент создается и добавляется в DOM;
- Updating (обновление) – компонент находится в DOM, обновление вызывается при изменении свойств или состояния компонента;
- Unmounting (демонтирование) – компонент удаляется из DOM.

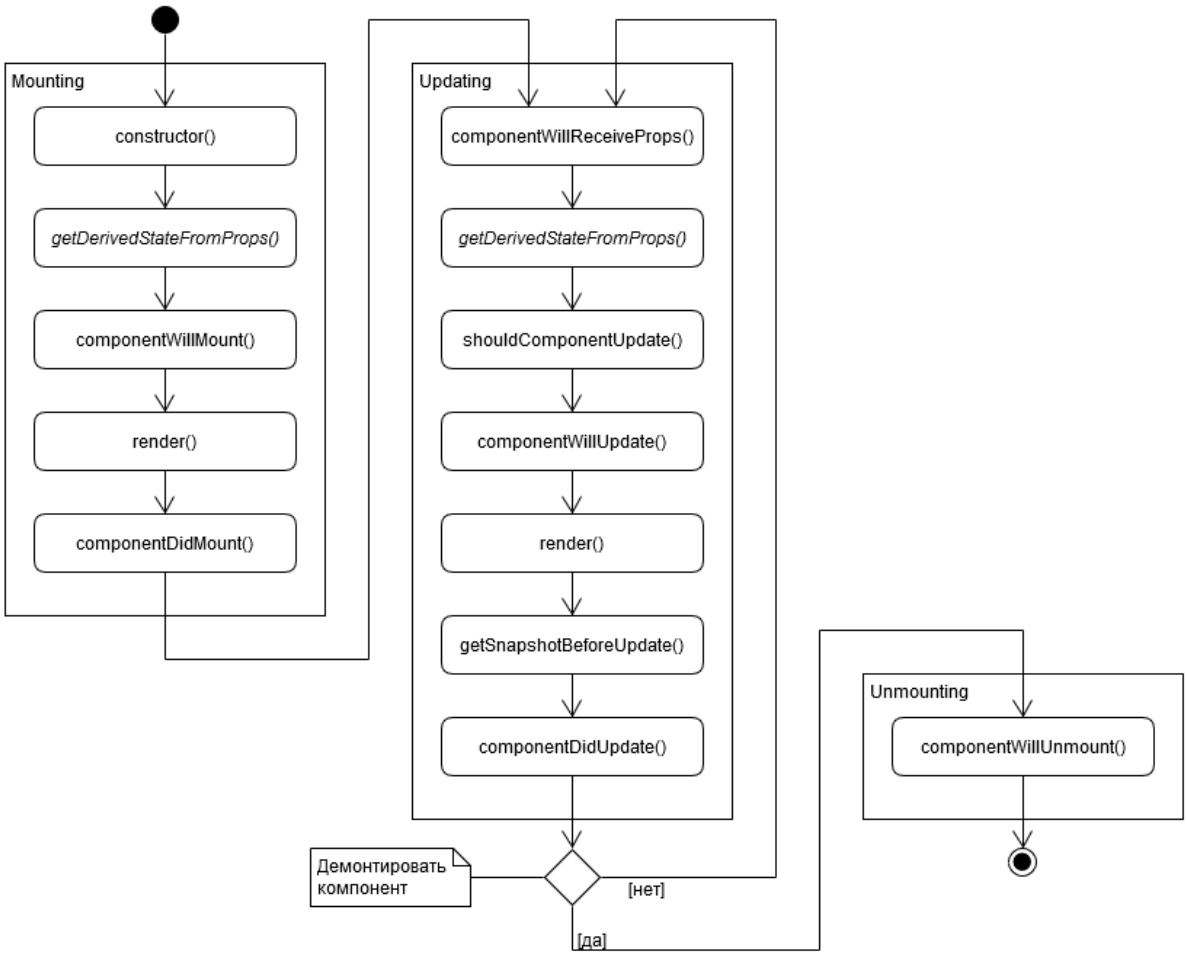


Рисунок 10 – Диаграмма деятельности жизненного цикла React-компонента

Помимо библиотеки React были выбраны следующие инструменты для создания клиентской части приложения:

- React Router – библиотека для работы с маршрутизацией компонентов, позволяет устанавливать соответствие между определенным адресом и React-компонентом, который должен быть отображен по этому адресу [11], удобна для создания одностраничного приложения, основанного на React;

- Bootstrap – набор инструментов, включающий в себя HTML- и CSS-шаблоны оформления пользовательского интерфейса [12].

2.3 Модель

Модель приложения была составлена на основе модели предметной области, описанной выше в разделе 1.3. Диаграмма ее классов представлена на рисунке 11. Поля каждого класса в действительности не публичны, доступ к ним предоставляется свойствами, на диаграмме они показаны публичными для компактности.

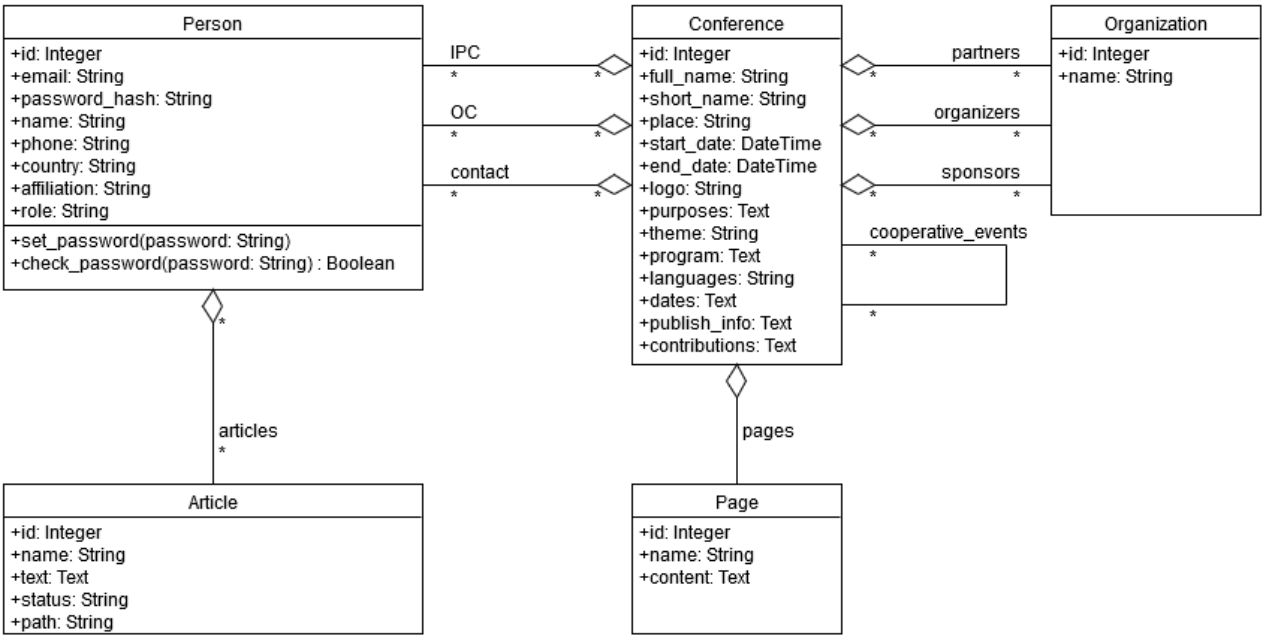


Рисунок 11 – Диаграмма классов модели приложения

В модели поля каждого класса соответствуют атрибутам множеств сущностей, которые будут созданы в базе данных в процессе миграции. В каждом классе есть поле id, которое соответствует суррогатному ключу в соответствующей таблице базы данных.

Пояснение к классу Person: в базе данных не хранятся пароли пользователей в чистом виде, пароль задается с помощью метода set_password(). При задании пароля к нему применяется хеш-функция, возвращающая некоторое значение (хеш). Этот хеш сохраняется в поле password_hash и хранится в базе в таком виде. Для проверки пароля в процессе аутентификации пользователя используется метод check_password(), который применяет к введенному пользователем паролю хеш-функцию, использованную для генерации соответствующего хеша, и возвращает значение True, если хеши совпали.

2.4 Контроллеры

Контроллеры приложения показаны на рисунке 12 в виде диаграммы классов.

Методы без параметров должны вызываться при получении GET-запроса, а с параметрами – при получении POST-запроса. Каждый метод возвращает ответ в виде JSON-объекта, преобразованного к базовому типу String.

Каждому классу модели соответствует класс-контроллер. Такие классы однотипны: они имеют набор CRUD-методов для работы с объектами модели. Каждый метод соответствует определенному адресу, так, например, если отправить запрос по адресу http://<адрес_сервера>/conferences/get_all, то вызовется метод conferences.get_all(), который вернет ответ в виде JSON-объекта со списком всех конференций, которые есть в базе.

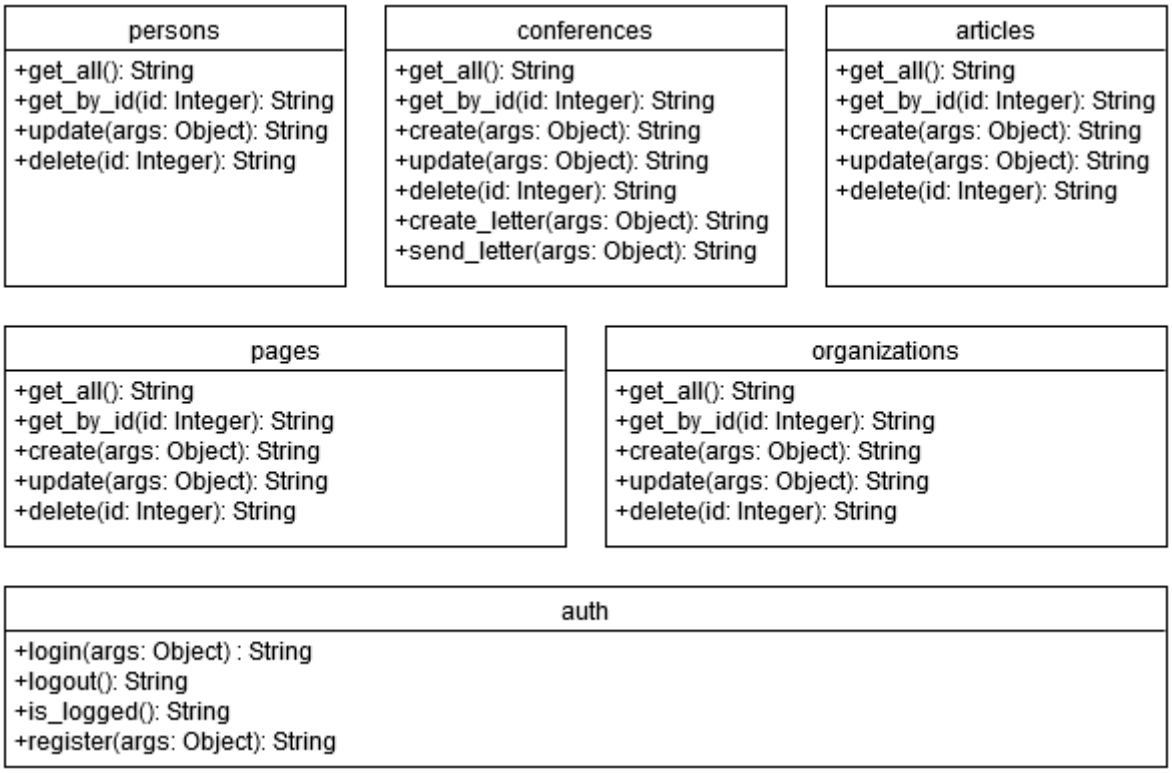


Рисунок 12 – Диаграмма классов контроллеров приложения

Помимо CRUD-методов класс conferences включает:

- create_letter() – формирует информационное письмо;

- send_letter() – отправляет информационное письмо;

Класс persons не содержит метода create() потому, что объект класса Person в системе может быть создан только в результате регистрации, которая происходит с помощью метода auth.register().

Класс auth включает в себя методы для работы с аутентификацией пользователей.

Все эти методы также возвращают ответ при возникновении ошибок. Так, например, при попытке изменить объект класса Conference (то есть вызвать метод conferences.update()) будут произведены проверки: существует ли объект с запрошенным id в базе, существует ли объект со значением уникального атрибута, равным соответствующему ему значению, пришедшему с запросом, и в случае возникновения ошибки метод вернет ответ с кодом этой ошибки.

2.5 Представление

На рисунке 13 показаны классы представления приложения.

Все классы представления унаследованы от класса Component библиотеки React. Класс (React-компонент) App – входная точка приложения, его экземпляр является

родителем (в контексте DOM) остальных компонентов страницы. При загрузке этого компонента выполняется запрос к серверной части для получения списка конференций, который затем сохраняется в состоянии компонента App. Кроме того, в его состоянии хранится информация о выбранной пользователем конференции.

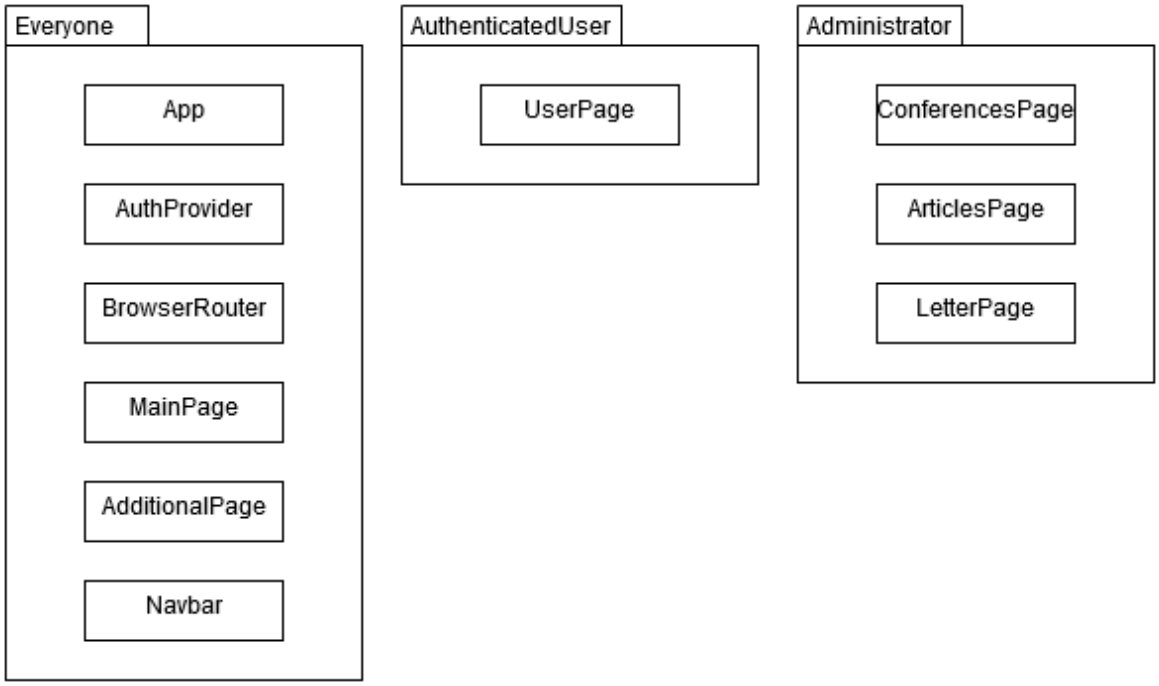


Рисунок 13 – Диаграмма классов представления приложения

AuthProvider – компонент, ответственный за аутентификацию пользователя. При загрузке этот компонент запрашивает у сервера, аутентифицирован ли пользователь в системе, и сохраняет ответ сервера в своем состоянии.

BrowserRouter – компонент из библиотеки React Router, используется для маршрутизации: при переходе на другую страницу приложения по ссылке React Router перехватывает следование по ссылке и заменяет предыдущий компонент другим компонентом, соответствующим адресу ссылки.

Остальные компоненты были разделены на три пакета: Everyone (доступны всем пользователям), AuthenticatedUser (доступны авторизированным пользователям) и Administrator (доступны только администраторам).

MainPage – компонент, соответствующий главной странице сайта, на которой отображается выбранная конференция (если конференция не выбрана, то будет отображена конференция по умолчанию).

AdditionalPage – отображает дополнительные страницы выбранной конференции.

Navbar – навигационное меню, предоставляет выбор конференции и форму входа/регистрации, а также инструменты администрирования для администратора.

UserPage – страница пользователя, на которой отображается информация о нем.

ConferencesPage – страница управления конференциями.

ArticlesPage – страница со списком научных статей выбранной конференции.

LetterPage – страница формирования и рассылки информационного письма.

На рисунках 14-16 показаны макеты главной страницы приложения (на рисунках 15, 16 показаны только те части макета, которые отличаются от макета на рисунке 14).

Рисунок 14 – Макет главной страницы (неавторизированный пользователь)

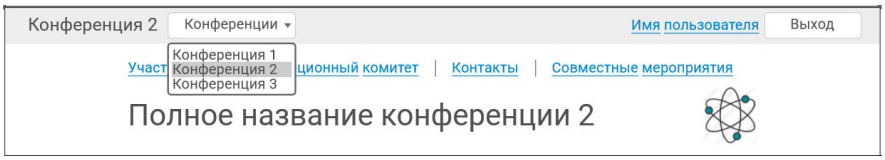


Рисунок 15 – Макет главной страницы (авторизированный пользователь)

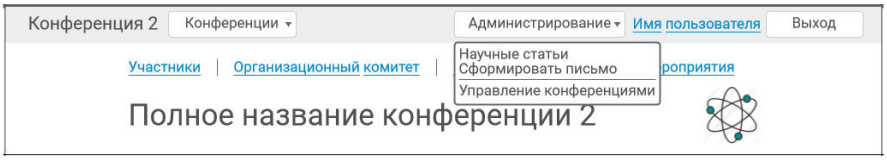


Рисунок 16 – Макет главной страницы (администратор)

3 Реализация

3.1 Серверная часть

Для реализации модели приложения были написаны ее классы, каждый из которых унаследован от класса Model библиотеки SQLAlchemy. Для реализации связей один-ко-многим и многие-ко-многим классам были добавлены поля типа relationship библиотеки SQLAlchemy, эти поля связывают таблицы по первичному ключу. Также для связей многие-ко-многим были описаны дополнительные таблицы. Класс Person был также унаследован от класса UserMixin библиотеки flask-login – это необходимо для контроля аутентификации пользователей. Затем с помощью библиотеки flask-migrate были созданы файлы миграции, на основе которых была составлена схема базы данных и создан файл самой базы данных.

Каждый класс-контроллер унаследован от класса Blueprint фреймворка Flask, это позволяет структурировать его методы, вызываемые при получении запросов к этому контроллеру, и избежать повторения кода. Каждому контроллеру добавляется поле url_prefix, которое является входной точкой для его методов. С помощью шаблона проектирования «Декоратор» методам сопоставляется определенный url, по запросу на который этот метод будет вызван. Каждому url может соответствовать только один метод, но методу может соответствовать несколько url, при запросе на каждый и которых он также будет вызван. Затем все контроллеры были зарегистрированы в приложении с помощью метода Flask.register_blueprint().

Работу CRUD-методов контроллеров рассмотрим на примере метода conferences.update() (рисунок 17, здесь и далее на диаграммах последовательности опущено преобразование возвращаемого значения к типу String):

- параметры (args) приходят в теле POST-запроса, отправленного по адресу http://<адрес_сервера>/conferences/update;

- функция Request.get_json() преобразует полученные параметры к базовому типу dictionary языка Python;

- проверяется, есть ли в параметрах id конференции, которую пользователь хочет поменять, и если нет, то возвращается ошибка;

- в ином случае вызывается метод Conference.query.get(), который обращается к базе данных с SQL-запросом (это задача модели, поэтому на диаграмме не показан этот SQL-запрос), этот метод вернет объект класса Conference при успешном результате запроса к БД, либо вернет значение None, если объект не найден в базе;

- проверяется, существует ли в базе конференция с указанным id, если нет, то возвращается ошибка;

- так как атрибут short_name является уникальным в БД, необходима дополнительная проверка, если в базе найдена хоть одна конференция с тем же именем, что пользователь хочет задать выбранной конференции, то возвращается ошибка;

- для каждого аргумента, переданного в теле запроса, соответствующему полю конференции присваивается значение этого аргумента;

- изменения сохраняются в базе данных;

- в случае возникновения ошибки при записи в БД возвращается ошибка;

- в случае успешной записи в БД возвращается положительный ответ.

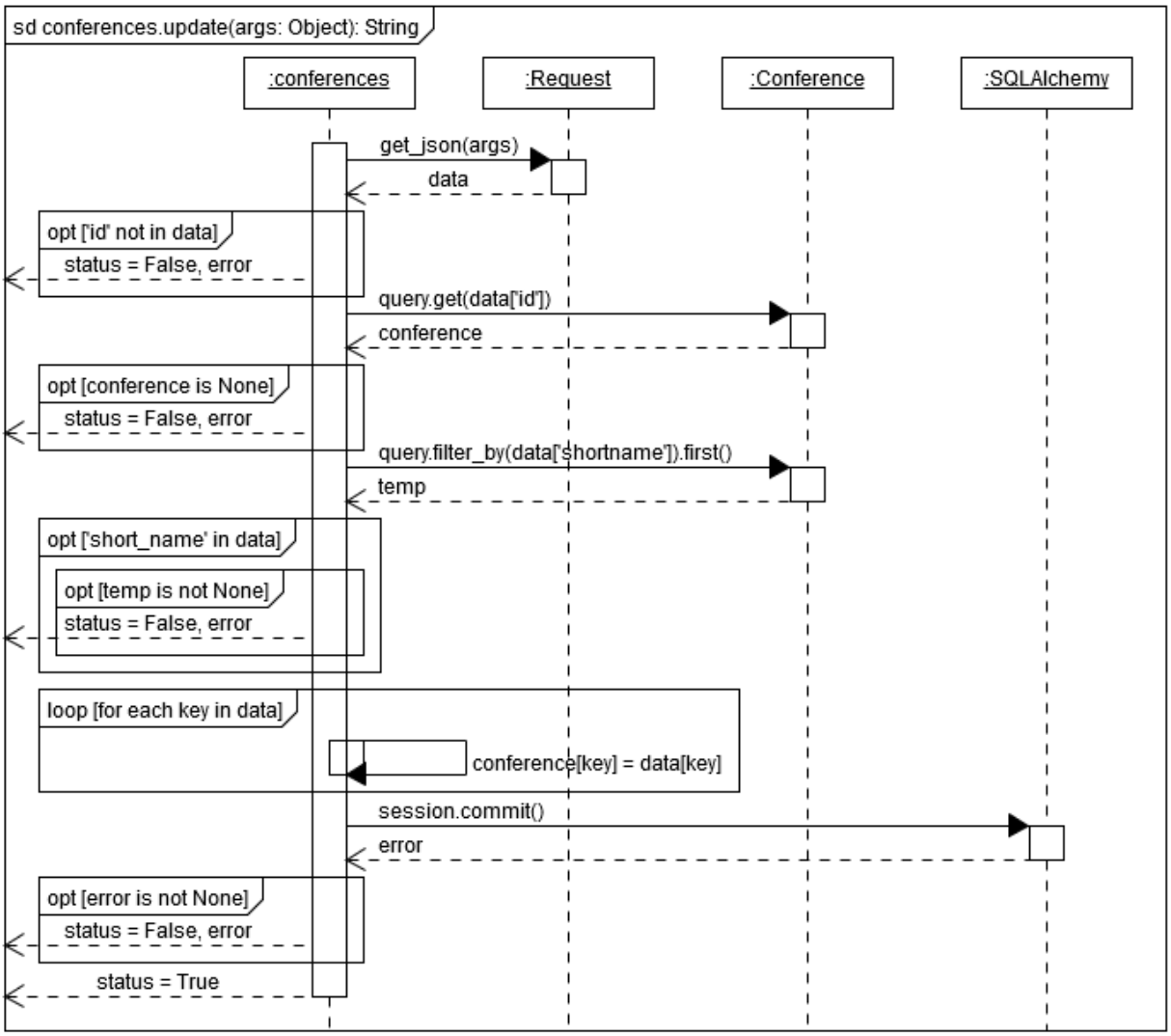


Рисунок 17 – Диаграмма последовательности метода conferences.update()

На рисунке 18 показана диаграмма последовательности метода conferences.create_letter(), в этом методе используется вспомогательный класс Letter, который не относится к модели, так как информационные письма не хранятся в базе, а формируются на основе конференций.

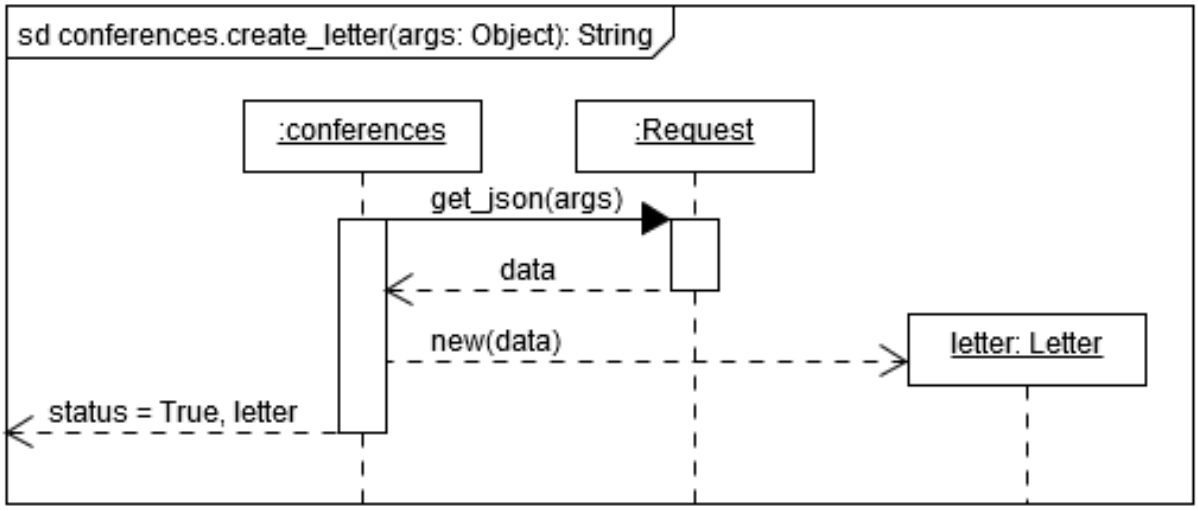


Рисунок 18 – Диаграмма последовательности метода conferences.create_letter()

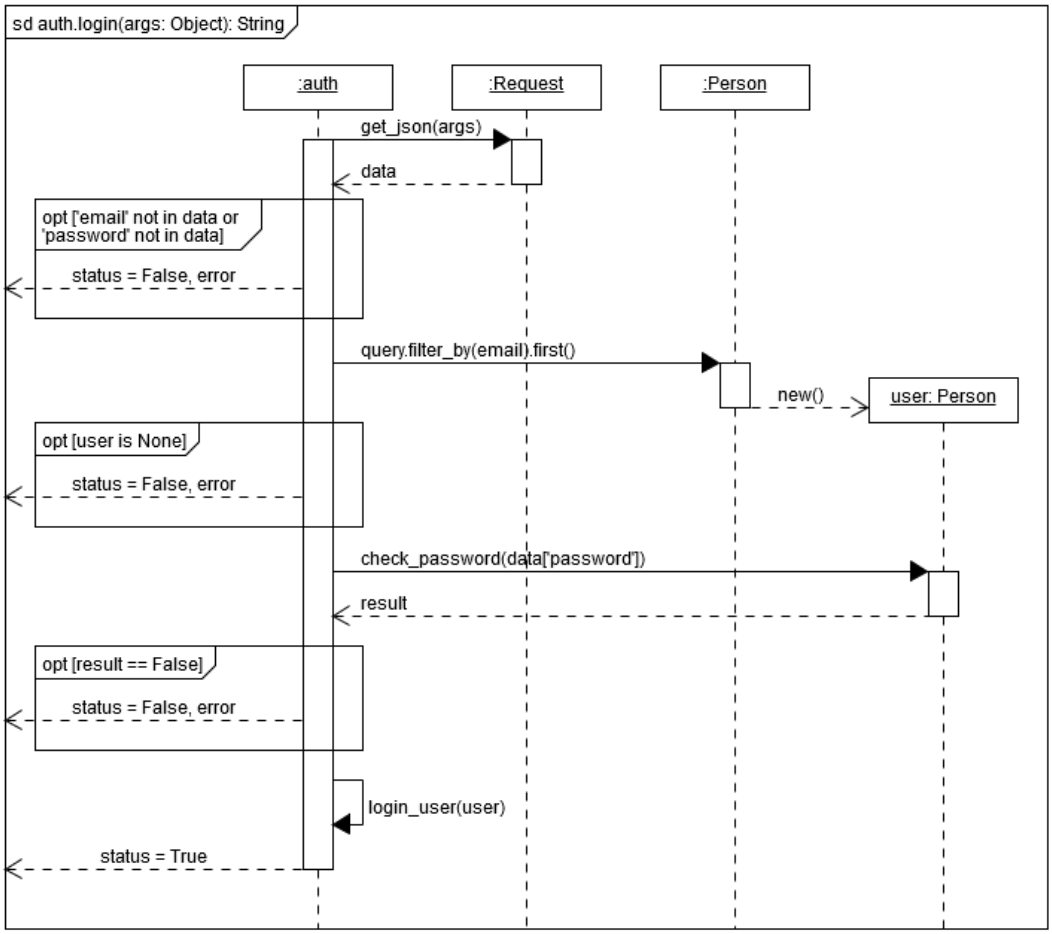


Рисунок 19 – Диаграмма последовательности метода auth.login()

На рисунке 19 представлена диаграмма последовательности метода auth.login(). Здесь производятся проверки пришедших с запросом данных, существует ли такой пользователь в базе и соответствует ли введенный пароль хешу пароля в базе. Затем вызывается функция login_user() из библиотеки flask_login и отправляется ответ.

Метод auth.is_logged() проверяет, авторизован ли текущий пользователь в системе, и если да, то возвращает status = True, его id и email, а если нет, то возвращает status = False.

Метод auth.logout() вызывает функцию logout_user() библиотеки flask_login и возвращает status = True.

Диаграмма последовательности метода auth.register() показана на рисунке 20.

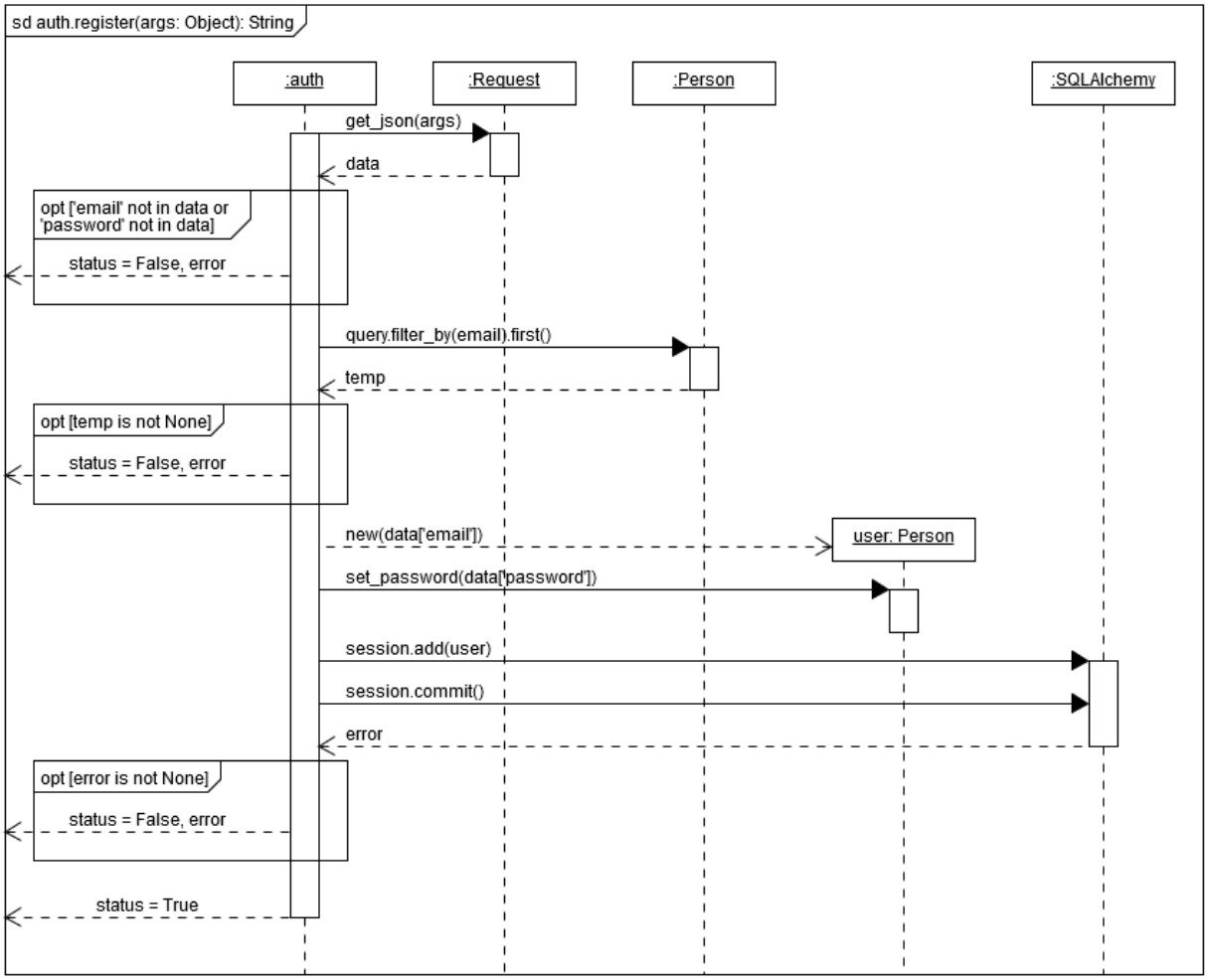


Рисунок 20 – Диаграмма последовательности метода auth.register()

Здесь проверяется, пришли ли данные, существует ли пользователь с совпадающим email. Затем создается объект user класса Person, ему задается email, пришедший с запросом, вызывается метод user.set_password(), далее объект сохраняется в базе с помощью SQLAlchemy и возвращается ответ.

3.2 Клиентская часть

В клиентской части приложения были реализованы классы, показанные на рисунке 13 в разделе 2.5. Далее на рисунках 21-23 показаны скриншоты приложения.

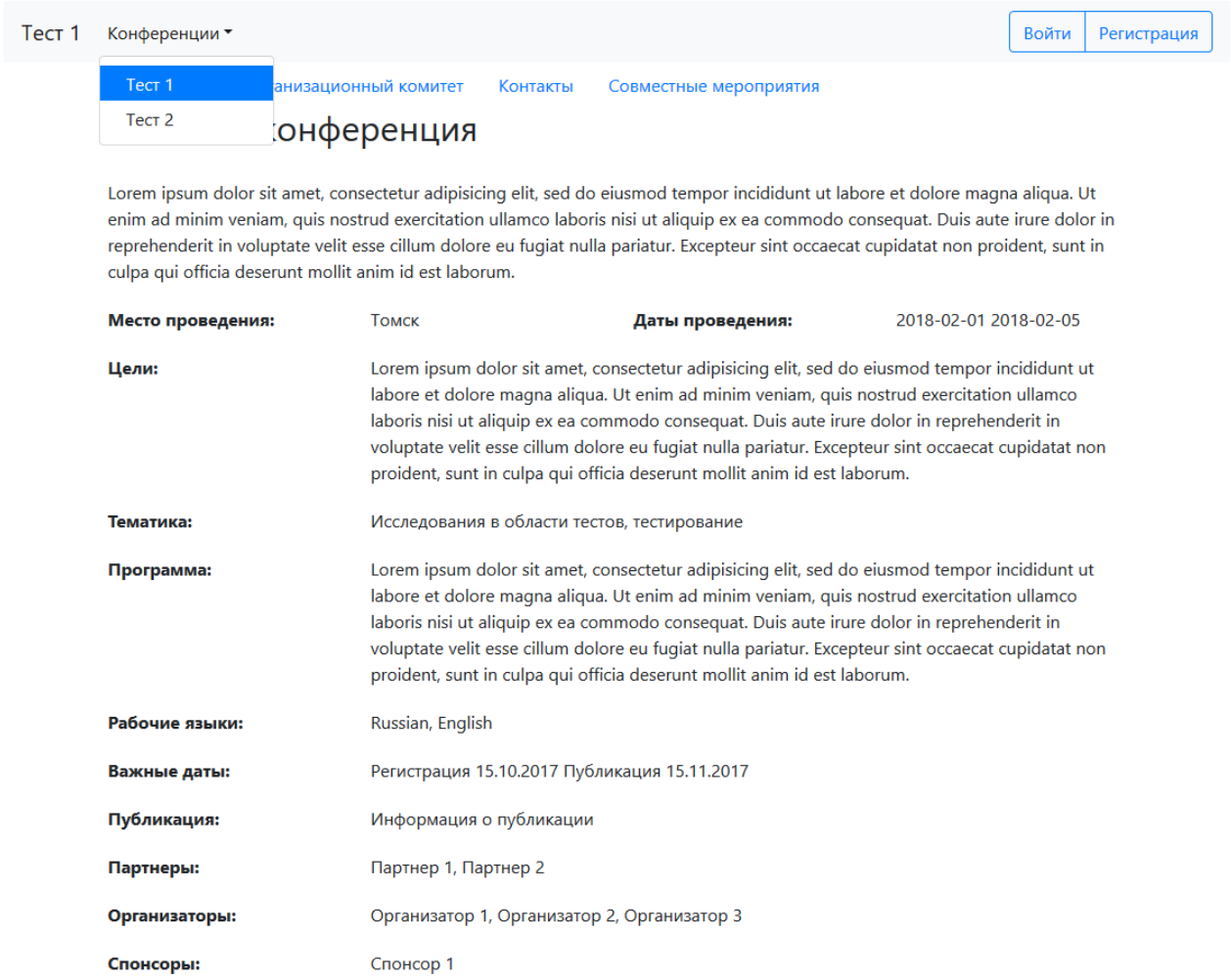


Рисунок 21 – Скриншот главной страницы (пользователь не авторизован)

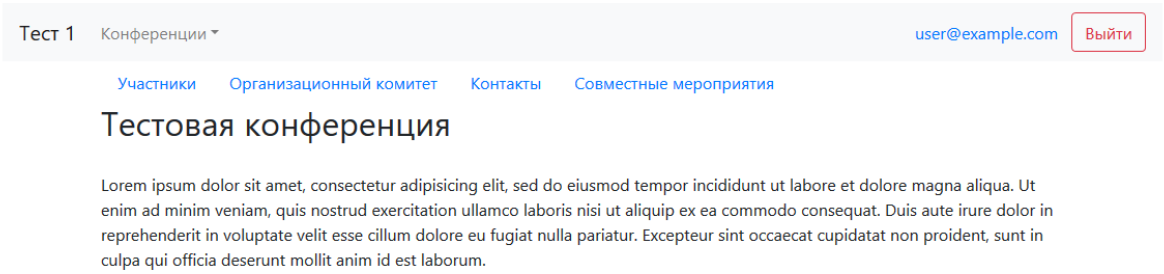


Рисунок 22 – Скриншот главной страницы (пользователь авторизован)

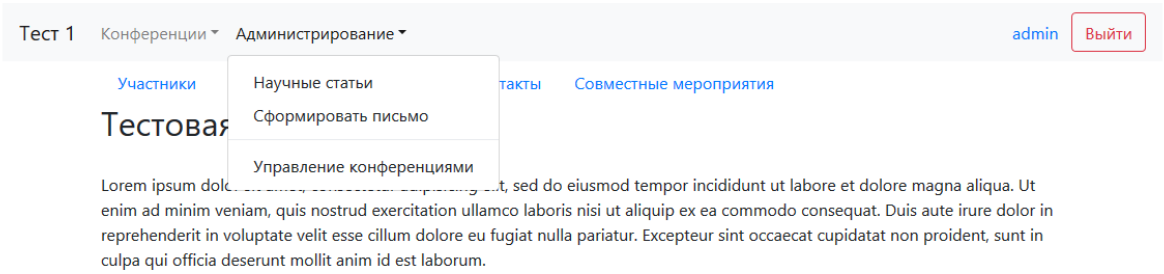


Рисунок 23 – Скриншот главной страницы (администратор)

На каждой странице отображается меню, на котором находятся:

- краткое название выбранной конференции (либо первой в списке, если пользователь не выбрал конференцию);

- раскрывающийся список конференций;

- список инструментов для администрирования, если текущий пользователь является администратором;

- форма авторизации/регистрации, либо ссылка на страницу авторизированного пользователя и кнопка выхода.

На главной странице сайта отображается информация о конференции.

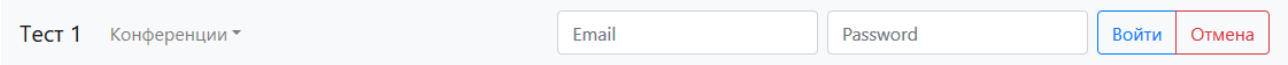


Рисунок 24 – Скриншот формы входа

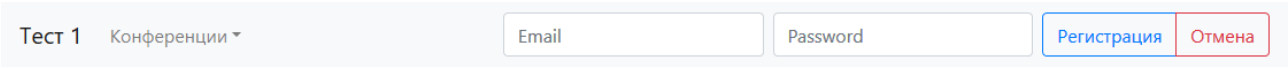


Рисунок 25 – Скриншот формы регистрации

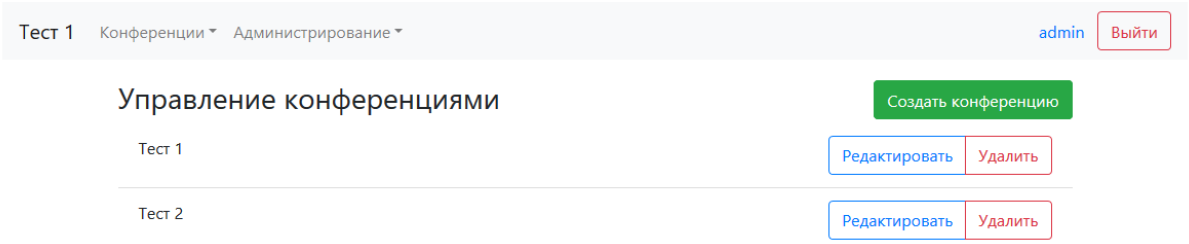


Рисунок 26 – Скриншот страницы «Управление конференциями»

На странице «Управление конференциями» отображается список всех конференций и элементы для их создания и редактирования.

ЗАКЛЮЧЕНИЕ

В процессе данной работы были решены все поставленные задачи:

- выявлены требования к системе;
- составлена модель предметной области;
- спроектирована архитектура системы;
- реализован проект.

Цель работы (разработать систему управления содержимым для сайтов научных конференций) достигнута.

ЛИТЕРАТУРА

1. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.]. – СПб: Питер, 2001. – 368 с.

2. Миковски М. Разработка одностраничных веб-приложений / М. Миковски, Д. Пауэлл. – М: ДМК Пресс, 2014. – 512 с.

3. Flask documentation. [Электронный ресурс] // URL: http://flask.pocoo.org/ (дата обращения: 15.05.2018).

4. PEP 3333 -- Python Web Server Gateway Interface v1.0.1 [Электронный ресурс] // URL: https://www.python.org/dev/peps/pep-3333/ (дата обращения: 15.05.2018).

5. Flask-Login documentation [Электронный ресурс] // URL: https://flask-login.readthedocs.io/en/latest/ (дата обращения: 15.05.2018).

6. SQLAlchemy – The Database Toolkit for Python [Электронный ресурс] // URL: https://www.sqlalchemy.org/ (дата обращения: 15.05.2018).

7. Flask-Migrate documentation [Электронный ресурс] // URL: https://flask-migrate.readthedocs.io/en/latest/ (дата обращения: 15.05.2018).

8. marshmallow documentation [Электронный ресурс] // URL: https://marshmallow.readthedocs.io/en/latest/ (дата обращения: 15.05.2018).

9. SQLite documentation [Электронный ресурс] // URL: https://www.sqlite.org/docs.html (дата обращения: 15.05.2018).

10. React - A JavaScript library for building user interfaces [Электронный ресурс] // URL: https://reactjs.org/ (дата обращения: 15.05.2018).

11. React Router: Declarative Routing for React.js [Электронный ресурс] // URL: https://reacttraining.com/react-router/ (дата обращения: 15.05.2018).

12. Bootstrap · The most popular HTML, CSS, and JS library in the world. [Электронный ресурс] // URL: https://getbootstrap.com/ (дата обращения: 15.05.2018).

Отчет о проверке на заимствования №1

Автор: g3304096@nwytg.com / ID: 5528187
Проверяющий: (g3304096@nwytg.com / ID: 5528187)

Отчет предоставлен сервисом «Антиплагиат»- http://www.antiplagiat.ru

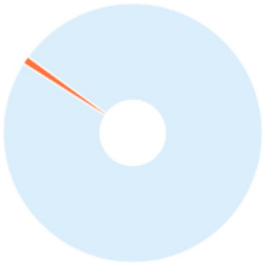
ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 3
Начало загрузки: 09.06.2018 17:19:22
Длительность загрузки: 00:00:11
Имя исходного файла: LauerDiploma
Размер текста: 834 кБ
Символов в тексте: 7222
Слов в тексте: 362
Число предложений: 47

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 09.06.2018 17:19:34
Длительность проверки: 00:00:00
Комментарии: не указано
Модули поиска:

ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
1,4%	0%	98,6%



Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Доля в тексте	Источник	Ссылка	Актуален на	Модуль поиска	Блоков в отчете	Блоков в тексте
[01]	1,4%	1,4%	Web Server Gateway Interface	http://en.wikipedia.org	13 Мар 2015	Модуль поиска Интернет	2	2
[02]	0%	1,4%	Web Server Gateway Interface	http://en.wikipedia.org	24 Апр 2017	Модуль поиска Интернет	0	2