Содержание

Введение

Целью работы является перенос сайта для просмотра студенческих работ на фреймворк Flask, а также интеграция в него реляционной СУБД и расширение функционала по поиску работ путем введения динамического тегирования текста курсовых работ.

Сайт должен предоставлять возможность студентам загружать работы на сайт, редактировать их, а также обеспечивать удобный поиск по базе работ и возможность составления отчетов. Весь функционал должен быть доступен только авторизованным пользователям. Авторизация должна проходить по учетным данным учебного сервера, через сервер LDAP. Серверная часть сайта должна быть выполнена на фреймворке Flask. Для реализации проекта требуется изучить:

- Авторизацию через LDAP на фреймворке Flask.
- Работу с базой данных через Python.
- Загрузку и выгрузку файлов с LDAP сервера.
- Редактирование html-разметки с помощью шаблонизатора Jinja.
- Варианты реализации информационного поиска по ключевым словам.

1. Актуальность

Обзор существующего решения:
На данный момент серверная часть сайта реализована на Closure. Closure – язык функционального программирования с ориентацией на многопоточность и компиляцию байт-кода под JVM и .NET платформы. На данный момент для поиска работ используется иерархическая структура файлового хранилища, а также файлы конфигурации с соответствующими данными студентов. Внедрение СУБД поможет оптимизировать процесс поиска работ. Также гибкость и возможности Python помогут реализовать новый функционал по информационному поиску в системе с применением NLP методов? и реализовать покрытие кода тестами, что упростит его дальнейшую поддержку. Общий стек технологий для всех сервисов института упростит взаимодействие между ними, а также их поддержку, что, несомненно, является преимуществом наряду с высокой популярностью и модульностью фреймворка Flask. Внедрение подсказок при вводе поможет улучшить пользовательский опыт при работе с сервисом.
На данный момент на сайте реализован основной функционал по поиску и загрузке курсовых работ в виде PDF. Пользователь может отыскать необходимую работу, зная нужный год, кафедру, группу, фамилию студента и преподавателя. Однако отсутствует поиск по темам или содержанию самих работ, что существенно облегчило бы процесс поиска нужной работы. Таким образом с расширением функционала сервиса появляется задача информационного поиска.

2. Спецификация требований

2.1. Функциональные требования

1. Авторизация через сервер ldap под учетными данными студентов.
2. Регистрация и редактирование учетных данных курсовой работы(название, данные руководителя, ФИО студента, тема, кафедра).
3. Поиск работ по критериям (группа, год, кафедра, руководитель,фамилия студента).
4. Возможность группировки по годам, кафедрам и группам.
5. Возможность загрузки и выгрузки отчета в формате pdf.
6. Составление отчетов о загруженных работах.
7. Реализация поиска работ по ключевым словам.

Ограничения:

1. Серверная часть сайта должна быть написана на фреймворке Flask.
2. Шапка сайта должна содержать необходимые ссылки на источники.
3. Клиентская часть не должна сильно отличаться визуально от первоисточника и других сайтов платформы института.

2.2. Детальные требования к основным подсистемам

Подсистема авторизации
Просмотр содержимого сайта доступен только авторизованным пользователям.
Реализация подсистемы авторизации через ldap сервер.
Авторизация через учетные данные студентов с учебного сервера.

Подсистема регистрации работ
Отображение информации об актуальной курсовой за текущий учебный год при открытии страницы.
Возможность просмотра и редактирования работ за предыдущие года.
Переход на страницы к курсовым прошлых лет (при наличии) осуществляется через пользовательский интерфейс путем выбора года в выпадающем списке или переходом по генерируемой гиперссылке.
На странице должна отображаться актуальная информация о ФИО авторизованного студента, его группе и курсе.
Возможность скопировать данные о курсовой за прошлый год (при наличии) путем нажатия соответствующей кнопки.

Подсистема загрузки работ
По-умолчанию открывается страница с загрузкой файлов для актуальной работы за текущий учебный год.
Для перехода к загрузке файлов за прошлые года необходимо реализовать выпадающий список с годами, на которые имеются зарегистрированные работы(либо гиперссылка).
Загрузка/ скачивание работ доступно только авторизованным пользователям.
Проверка файла на соответствие формату PDF.
Форма загрузки должна иметь следующие поля:

- ФИО руководителя (поле для ввода текста)
- Должность руководителя (выпадающий список)
- Научная степень (поле для ввода текста)
- Ученое звание(выпадающий список)
- Кафедра(выпадающий список)
- Номер группы(выпадающий список)

После загрузки файла, должно производится считывание текста работы и генерация тегов.
Доступные поля для загрузки файлов отличаются в зависимости от курса студента.
Для всех студентов:

- Промежуточный отчет
- Промежуточная презентация
- Отчет

- Презентация

Для студентов выпускных курсов предусмотрены доп. поля:

- Отзыв руководителя
- Проверка на плагиат

Для студентов выпускных курсов магистратуры предусмотрены доп. поля:

- Рецензия

Подсистема просмотра работ

Форма поиска работ должна иметь следующие поля:

Обязательные для заполнения:

- Номер группы(выпадающий список, доступен множественный выбор)
- Год (выпадающий список, доступен множественный выбор)

Необязательные для заполнения:

- Кафедра (выпадающий список)
- Студент (поле для ввода текста)
- Руководитель (поле для ввода текста)
- Тема (поле для ввода текста)
- Группировка (выпадающий список)
- Сортировка(выпадающий список)
- Контекстный поиск(поле ввода текста)

Таблица с курсовыми должна иметь следующие поля для студентов всех курсов:

- Номер
- Тема
- Студент
- Руководитель
- Промежуточный отчет
- Промежуточная презентация
- Отчет
- Презентация

Таблица с курсовыми должна иметь дополнительные поля для студентов выпускных курсов:

- Отзыв руководителя
- Проверка на плагиат
- Отчет по практике НИР

Таблица с курсовыми должна иметь дополнительные поля для студентов выпускных курсов:

- Рецензия

Для каждой кафедры/года/группы должна формироваться отдельная таблица.

Напротив каждой группы, кафедры, года должно указываться количество соответствующих записей.
Ячейки таблицы должны содержать ссылки на скачивание необходимых pdf-файлов.
Ячейки со ссылками на работы, загруженные ранее установленного срока должны иметь белую заливку, соответственно ячейки с отчетами, загруженными позже срока должны иметь красную заливку.

Поле для контекстного поиска должно отображать пример запроса в качестве подсказки. Для набора текста в поле поиска должна быть реализована функция помощи ввода на основании существующих тегов, реализованная в виде выпадающего списка.

Подсистема тегирования текста
Должен считываться весь текст отчета, за исключением титульного листа и списка источников.
Генерация тегов не должна занимать более 10 секунд для каждой работы, допустимо подгружать теги после загрузки самой работы.
Теги не должны содержать предлоги, союзы, местоимения, прилагательные и другие общеупотребительные слова, а также числа.
Сбои в работе подсистемы реферирования не должны сказываться на работе остальных систем. При ошибке генерации тегов, пользователь должен получать соответствующее уведомление, но загрузка работы должна завершаться корректно.

3. Информационный поиск

Было выделено несколько возможных вариантов для улучшения процесса поиска.

3.1. **Поиск по направлению работы.**

Изучение материала студенческих отчетов позволило выделить 13 основных направлений работ:

- нейросети и машинное обучение (классификация, кластеризация, регрессия, генерация, nlp)
- разработка веб-приложений,
- робототехника,
- мат. Моделирование и анализ,
- комп. Зрение и обработка изображений,
- IoT,
- чат боты,
- моб. Приложения,

- СППР,
- оптимизационные задачи,
- задачи позиционирования,
- комп. Графика,
- учетные системы

Возможным решением было указание студентом или руководителем подходящих направлений для каждой загружаемой работы вручную. Что в будущем упростило бы процесс поиска и позволило бы анализировать тенденции научных интересов в университете. Однако появляется проблема разметки уже загруженных работ, учитывая их количество, ручная разметка была бы весьма трудоемкой задачей. Но для этой задачи может использоваться многометочная классификация. В ручную размеченные работы, в таком случае, могут стать обучающей выборкой для нейронной модели, которая позволит классифицировать остальные работы.

3.2. **Генерация уникальных тегов для каждой работы.**

3.2.1. **Использование частотных словарей.**

Для поиска конкретной работы в списке можно использовать ключевые слова (далее теги). Каждый тег может представлять из себя уникальное слово, встречающееся в данной работе намного чаще, нежели в остальных. Теги не должны включать в себя местоимения, прилагательные и другие общеупотребительные слова, которые не помогут выделить конкретную работу из всего множества. Также вводимые пользователям теги, как и генерируемые при загрузке необходимо лемматизировать, чтобы исключить несоответствие из-за разной формы слова в поиске и базе данных. Для составления тегов можно использовать как и стандартные методы с использованием частотных словарей с фильтрами, так и нейросетевые модели для выделения ключевой информации из текста, с последующим составлением словаря. Классификация в данном подходе не подойдет,тк появляется необходимость заранее знать все возможные теги, характеризующие любую из работ. Таким образом были выделены следующие методы генерации тегов:

3.3. **Использование обученных моделей для реферирования текста с последующим составлением словаря**

Использование предобученных моделей для реферирования текста, способных хорошо справляться с поставленной задачей, требует больших вычислительных ресурсов и приведет к неудовлетворительной скорости обработки текстов для конкретной задачи. Вариант решения – сократить объем анализируемого текста до введения и заключения.

3.4. Использование Term Frequency-Inverse Document Frequency методов

Данный метод определяет важность конкретного слова в контексте целого корпуса текстов.

В nlp-модуле из библиотеки sklearn.feature_extraction.text есть готовая реализация этого метода. Данный модуль предназначен для составления списка ключевых слов, описывающих текст. Для обработки метод также удаляет общеупотребительные слова из текста, т.к. они не несут особой смысловой нагрузки и не могут являться ключевыми в данном случае.

Метод анализирует как частоту встречаемости слова внутри самого текста, так и частоту данного слова в остальных текстах, на основании этого определяет веса для каждого слова. Данные хранятся в виде матрицы, где каждая строка матрицы определяет строку (текст), а столбец – само слово. Ячейка матрицы – вес слова в конкретной строке (тексте). Можно обучить одну общую матрицу для тегирования большинства слов в тексте и при необходимости дооубучить ее на новых данных из загружаемых работ. Но библиотека работает по умолчанию только с английским языком, однако можно также, загрузить русский язык, используя модуль из библиотеки nltk. Работа метода заключается в 3 этапах.

3.4.1. TF – Term Frequency

Оценивается относительная частота встречаемости слова в документе.

3.4.2. IDF – Inverse Document Frequency

Оценивается важность слова в контексте корпуса текстов. Слова, которые встречаются в других текстах чаще будут иметь более низкое значение IDF, вычисляемое как:

$$IDF=\log(N/m), \text{ где}$$

N– общее количество текстов,

m– количество текстов, содержащих данное слово

3.4.3. TF-IDF

Итоговое значение TF-IDF= TF*IDF, определяет важность слова в текущем тексте, учитывая контекст из всего корпуса текстов, следовательно слова с наибольшим значением TF-IDF будут являться уникальными тегами для конкретной работы.

Однако метод имеет следующие недостатки:

- Влияние размера текста, на точность метрики.

- Снижение веса слов/терминов из специфических областей
  В случаях, когда в выборке текстов часто встречается слово, относящееся к определенной области знаний, может произойти снижение веса этого слова в общей матрице, несмотря на то, что слово может быть важным контексте научной работы.

3.5. **Использование ключевых терминов предметной области в качестве тегов.**
Большая часть научных работ содержит глоссарий специфических терминов, характерных для конкретной предметной области. Часть этих терминов, с высокой вероятностью поможет выделить конкретную работу из общего множества. Искать глоссарий можно с использованием регулярных выражений, также при поиске повышенный приоритет следует уделять первым страницам, которые как правило содержат определения ключевых терминов.

3.6. Вариант комбинации из перечисленных методов

4. Проектирование

4.1. Высокоуровневая архитектура

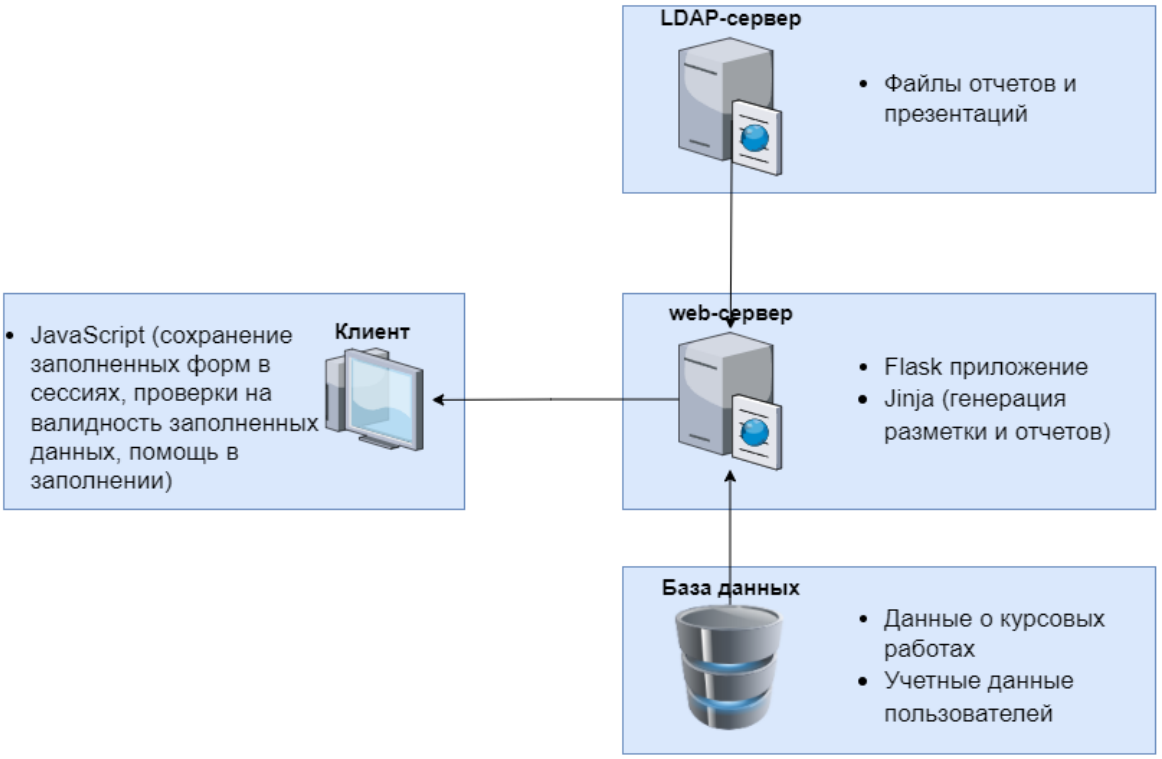

Рис. 1 – высокоуровневая архитектура

4.1.1. Диаграмма классов

{новая диаграмма классов}

Рис. 2 – диаграмма классов

4.1.2. Реляционная модель

CourseWork			
Имя	Определение	Тип	Ключи
Topic	Тема работы	varchar(200)	-
Department	Номер кафедры	int	-
Head	Руководитель	varchar(200)	-
First report	Ссылка на промежуточный отчет	varchar(200)	-

Report	Ссылка на отчет	varchar(200)	-
Presentation	Ссылка на презентацию	varchar(200)	-
First presentation	Ссылка на презентацию	varchar(200)	-
ID	Идентификатор записи	int	PK
student ID	Айди студента	int	FK(student)
date	Дата публикации	date	-
group	Группа студента на момент публикации	String	-

User			
Имя	Определение	Тип	Ключи
Name	Имя	varchar(200)	-
Surname	Фамилия	varchar(200)	-
Patronymic	Отчество	varchar(200)	-
IsStudent	Студент/преподаватель	bool	-
Login	Логин	varchar(200)	-
Password	Пароль	varchar(200)	-
ID	Идентификатор пользователя	int	PK

Head			
Имя	Определение	Тип	Ключи
DepartmentID	ID Кафедры	int	FK(departments)
Degree	Научная степень	varchar(200)	-
SinceStatus	Научное звание	varchar(200)	-
Status	Должность	varchar(200)	-

ID	Идентификатор преподавателя	int	PK
userID	Идентификатор пользователя	int	FK(user)

Student			
Имя	Определение	Тип	Ключи
Group	Группа	varchar(200)	-
ID	Идентификатор студента	int	PK
userID	Идентификатор пользователя	int	FK(user)

4.2. Сценарии использования

{Диаграммы и текстовое описание для сценариев работы с каждой подсистемой.}

4.2.1. Просмотр курсовый работ

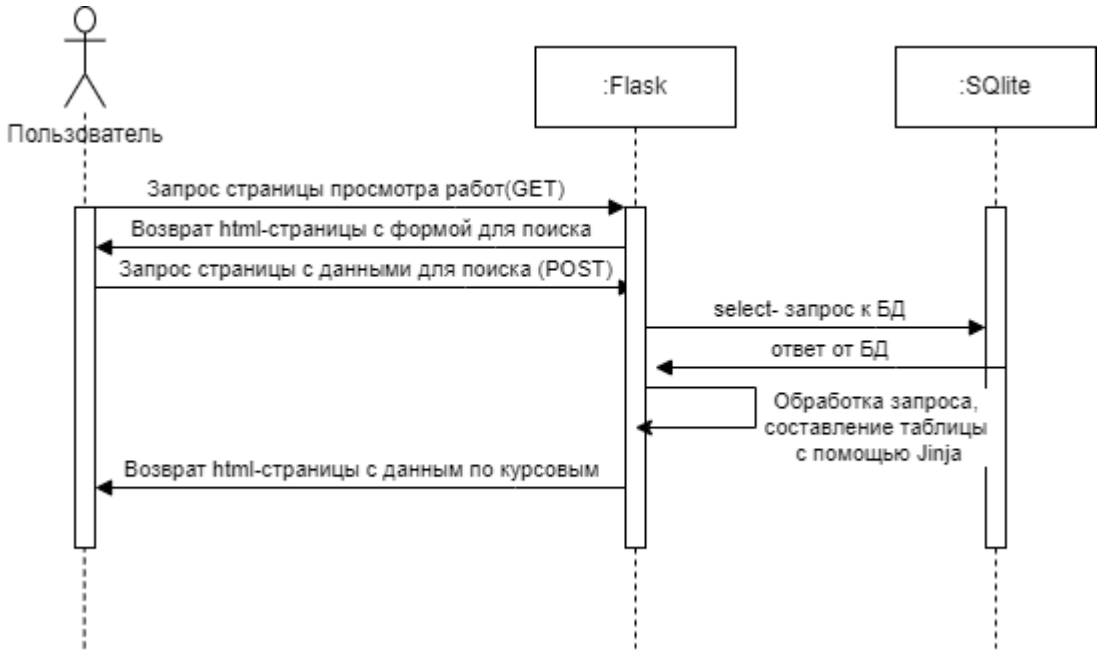

Рис. 3 – диаграмма последовательности для страницы просмотра работ

4.2.2. Загрузка курсовой работы

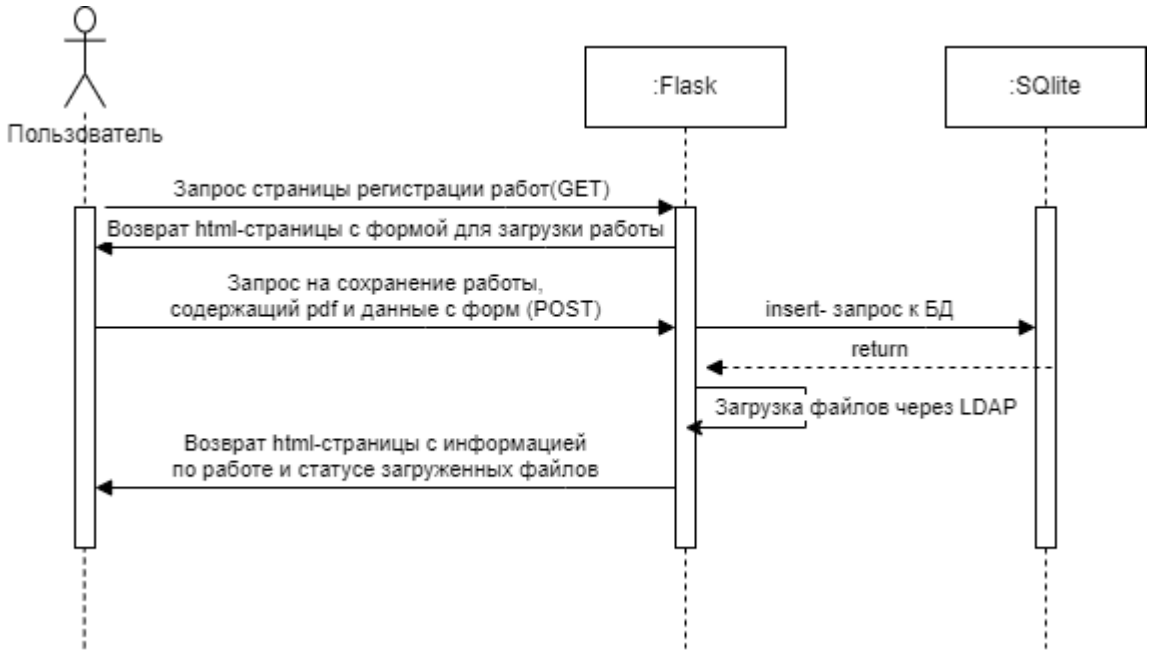

Рис. 4 – диаграмма последовательности для страницы загрузки работ

4.2.3. Регистрация курсовый работы

{Диаграммы и текстовое описание для сценариев работы с каждой подсистемой.}

4.3. Используемые инструменты

LDAP (Lightweight Directory Access Protocol) – это протокол для доступа к распределенным каталогам информации. LDAP-сервер обеспечивает централизованное хранилище информации о пользователях, группах, компьютерах и других объектах в сети.

Flask – это фреймворк для разработки веб-приложений на языке Python. Он является легковесным, но при этом достаточно мощным, чтобы создавать сложные приложения.

Одна из главных особенностей Flask – его модульность. Он содержит самые необходимые функции, а все остальные могут быть добавлены через расширения. Это позволяет разработчикам создавать приложения с учетом тех функций, которые им нужны, что делает Flask очень гибким.

Flask обладает широким сообществом пользователей и подробной документацией. Flask – это отличный выбор для начинающих разработчиков, а также для более опытных, которые хотят создавать быстрые и масштабируемые веб-приложения.

SQLite – реляционная база данных, которая хранит данные в локальном файле без использования отдельного сервера баз данных. Она позволяет создавать и управлять малыми базами данных на компьютере или мобильном устройстве.

Основные преимущества SQLite заключаются в его портативности, простоте и быстродействии. SQLite работает без необходимости установки дополнительного ПО и подходит для любой операционной системе, обеспечивает высокую производительность при выполнении запросов, благодаря оптимизации работы с данными.

SQLite использует стандартный язык запросов SQL и обладает широким набором возможностей для создания таблиц, индексов и других объектов базы данных. SQLite поддерживает транзакции и контроль целостности данных, что делает ее надежной для хранения критически важных данных.

Из-за легковесности SQLite хорошо подходит для мобильных приложений, веб-приложений и других проектов с ограниченными требованиями к масштабируемости и производительности.

SQLAlchemy–это библиотека для языка Python, облегчающая работу с базами данных. Она позволяет разработчикам через Python API создавать объектно-реляционные отображения для баз данных и управлять ими.

Основным преимуществом SQLAlchemy является то, что она обеспечивает высокоуровневый интерфейс для работы с базами данных, что значительно упрощает процесс написания и поддержки кода. SQLAlchemy позволяет использовать SQL-запросы и объектно-ориентированные модели, это обеспечивает лучшую гибкость и масштабируемость приложений.

SQLAlchemy не зависит от конкретной СУБД, что позволяет переносить приложение на другую базу данных без необходимости изменять код. SQLAlchemy поддерживает множество реляционных баз данных, включая SQLite, MySQL, PostgreSQL, Oracle.

Объектно-ориентированные модели SQLAlchemy обеспечивают автоматическое преобразование данных между моделью приложения и таблицами базы данных, что делает процесс работы с данными более простым и интуитивно понятным. SQLAlchemy предоставляет мощный механизм миграции базы данных, который самостоятельно обновляет схему базы данных в соответствии с изменениями модели приложения.

SQLAlchemy мощный и гибкий инструмент для работы с базами данных на Python, который облегчает процесс написания и поддержки кода и позволяет создавать более надежные и масштабируемые приложения.

Jinja – это язык шаблонизатора для Python, который позволяет разработчикам легко создавать динамические HTML, XML, PDF и другие файлы. Он используется во многих фреймворках веб-разработки на Python, включая Flask и Django.

Главной особенностей Jinja является разделение логики приложения и представления, что повышает читаемость кода и ускоряет процесс разработки. Jinja позволяет использовать условные операторы, циклы, переменные и многое другое, чтобы создавать шаблоны с высокой степенью гибкости и адаптивности.

С помощью Jinja разработчики могут легко создавать и наследовать шаблоны, которые состоят из общих элементов, таких как шапка и подвал, и затем использовать эти элементы повторно на множестве других HTML страниц. Это позволяет значительно ускорить процесс разработки и обеспечить единообразие визуального оформления для всего сайта.

Finz - для работы с пфд файлами на python?
NLTK (natural language toolkit) для составления словарей тегов, исключения стоп-слов?
PyMorphy для лемматизации слов (тегирование)?
Skelearn для генерации тегов (тегирование)?

5. Разработка

5.1. Просмотр работ

За основу страницы была взята титульная web-страница с сайта kurs.cs.petrsu.ru. Для хранения записей была создана база данных SQLlite и таблица с записями о курсовых.
Страница отображается двумя способами. Если пользователь впервые открыл страницу и не указал данных для поиска, то отправляется GET-запрос на сервер, который возвращает стандартную страницу с полями по-умолчанию. Если пользователь ввел необходимые данные, то отправляется POST-запрос, содержащий данные из форм. Страница обновляется, для сохранение введенной информации в полях используется обращение к сессионному хранилищу в JavaScript. На сервере из запроса извлекаются данные из заполненных форм. С помощью объектно-ориентированного интерфейса

SQLAlchemy для Python формируется запрос к базе данных, на основе выбранных настроек и полей. Отправляется запрос к базе данных на выборку данных. Результат запроса, сгруппированный по годам и кафедрам в виде списка передается шаблонизатору Jinja для формирования таблиц с необходимыми данными и ссылками на отчеты в html-разметке. В коде для шаблонизатора прописаны шаблоны разметки и логика обхода списка для создания новых таблиц.
Таким образом реализован поиск по году, группе, кафедре, имени руководителя и студента, а также группировка по кафедрам и группам. См. рис 5.

Результат:
{картинка?}

Рис. 5 – страница просмотра работ

5.2. Регистрации работ

За основу страницы была взята web-страница с сайта kurs.cs.petrsu.ru. Для добавления записей реализованы основные insert-запросы к базе данных через интерфейс SQLAlchemy, также добавлена валидация данных на стороне сервера. Перед записью новых данных проверяется наличие актуальных записей за текущий год на серверной стороне, при их наличии создается серверный вызов с диалоговым окном, подтверждающим перезапись существующих данных.
{Здесь про логику регистрации .}

5.3. Загрузка работ

{Здесь про логику загрузки что-то.}

5.4. Тегирование работ

{Здесь про выбранные методы тегирования, библиотеки, логику.}

6. Тестирование

Каждый модуль должен иметь список позитивных и негативных тестов. Позитивные проверяют корректность работы системы на корректных данных, а негативные проверяют исключительные ситуации.
Тестирование должно проводиться как на валидность самих данных в выборке из базы данных, так и на соответствие отображаемых на сайте данных изначальной выборке. Тесты должны покрывать все возможные сценарии эксплуатации системы. Для проверки корректности отображения данных на сайте можно использовать генерацию запросов к серверу на выдачу html-документов. Затем производить проверку валидности результата, т.е. отсутствие как клиентских, так и серверных ошибок во время выполнения, наличие необходимых данных в полученной html-разметке, отсутствие лишних данных.
Покрытие тестами позволит проще обнаружить ошибки в коде, проверить работоспособность сервиса во всех сценариях и упростить дальнейшую поддержку сервиса.
Для запросов к серверу можно использовать библиотеку *request* для *python*, и *unittest* для проверки валидности ответов.
Примеры тестов:

6.1. Просмотр работ

	Позитивный тест	Негативный тест
Поиск работ по выбранному году и всем кафедрам	Выводятся все работы из тестовой выборки с нужным годом регистрации, проверка названий и общего количества записей	Не выводятся работ за другие года, пустых таблиц, при отсутствии записей в тестовой выборке выводится соответствующее оповещение
Поиск работ по году и кафедре	Выводятся все работы из тестовой выборки с нужным годом регистрации и кафедрой, количество записей в таблицах соответствует количество тем, подходящих по фильтрам и общего количества записей	Отсутствуют записи за другие года и кафедры
Поиск работ по неск. годам	Присутствуют заголовки с нужными годами, записи соответствуют выбранным	Отсутствуют работы за другие года

	фильтрам	
Поиск работ по неск. годам и группам с указанием	Присутствуют заголовки с нужными годами и группами, записи соответствуют выбранным фильтрам	Отсутствуют записи, не соответствующие фильтрам

6.2. Загрузка работ

{Список позитивных и негативных тестов.}

6.3. Регистрация работ

{Список позитивных и негативных тестов.}

6.4. Тегирование работ

{Здесь метрики тегирования, примеры какие теги указаны в поиске, какие нашлись, возможно сравнения разных методов.}

Заключение

{здесь про то, какие задачи выполнены, что изучено, что не получилось, что можно улучшить}

Список источников

1. Документация по Flask – Текст: электронный. – URL:https://flask.palletsprojects.com/en/latest/
2. Руководство по Flask – Текст: электронный. – URL: https://docs-python.ru/packages/veb-frejmvork-flask-python/
3. Авторизация через ldap – Текст: электронный. – URL: https://code.tutsplus.com/tutorials/flask-authentication-with-ldap--cms-23101
4. Работа с шаблонизатором Jinja – Текст: электронный. – URL: https://docs-python.ru/packages/veb-frejmvork-flask-python/rabota-shablonami-prilozhenii-flask/
5. Работа с базой данных – Текст: электронный. – URL: https://code.tutsplus.com/ru/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972
6. Библиотека ldap для Flask – Текст: электронный. – URL: https://pypi.org/project/Flask-LDAPConn/
7. Подключение SQLalchemy – Текст: электронный. – URL: https://pythonru.com/biblioteki/ustanovka-i-podklyuchenie-sqlalchemy-k-baze-dannyh
8. Учебник по Flask – Текст: электронный. – URL:https://habr.com/ru/articles/193242/
9. Документация на Flask (русский перевод) – Выпуск 0.10.1 – 2019 – 185 с. – Текст: непосредственный