Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.Б09-мм

Расширение функциональности Miminet: добавление автогенерации разделов тестов для тренажёра

Аржаев Дмитрий Вячеславович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
старший преподаватель кафедры системного программировани Зеленчук И. В.

Санкт-Петербург
2025

Оглавление

Введение

Проект Miminet [3], который используется в курсе «Компьютерные сети» на математико-механическом факультете СПбГУ, включает в себя интерактивный тренажёр, предназначенный для контроля знаний и закрепления изученного материала. Тренажёр содержит разнообразные тестовые задания, охватывающие ключевые темы дисциплины. Задания представлены в различных форматах, включая тесты с выбором ответов, практические задачи и симуляции сетевых процессов, что позволяет студентам не только проверить теоретические знания, но и применить их в условиях, приближенных к реальным.

Создание тестов осуществляется в административной панели вебсервиса Miminet. Тест поделен на разделы, в каждом из которых находятся вопросы, разбивающиеся на категории. Так, при создании вопроса обязательно указывается категория, к которой он принадлежит, и опционально раздел, который будет содержать этот вопрос. А при создания теста отмечаются разделы, которые будут в него входить.

Эта система задает заранее определенный набор вопросов и их порядок, которые не могут изменятся. То есть при любом прохождении раздела теста будут встречаться одни и те же вопросы в одинаковом порядке. Такой подход усложняет подготовку контрольного тестирования, поскольку список вопросов для каждого варианта статический и должен добавляться в раздел вручную. Кроме того, использование неизменяемого набора вопросов может повлиять на объективность оценки: студенты, заранее зная последовательность заданий, получают преимущество, что снижает достоверность результатов. Для решения этой проблемы необходимо модифицировать систему редактирования тестов, реализовав возможность случайного выбора вопросов из предварительно созданной категории заданий. Это позволит при каждом новом прохождении раздела формировать уникальный набор вопросов.

Для этого в редакторе разделов будет добавлено новое необязательное для заполнения поле, содержащее информацию для автоматической генерации списка вопросов по категориям.

1. Постановка задачи

Целью работы является расширение функциональности Miminet с помощью добавления возможности создания раздела теста с генерацией списка произвольных вопросов по заданной категории и количеству. Для ее выполнения были поставлены следующие задачи:

1. Дополнить интерфейс редактора разделов для ввода метаданных в админ-панели приложения Miminet

2. Добавить возможность генерации списка случайных вопросов при наличии метаданных

2. Обзор

Поскольку целью работы является расширение уже реализованной системы редактирования тестов, в обзоре существующих решений нет необходимости. Вместо этого акцент был сделан на анализе технологий, которые уже активно используются в проекте и на которых базируется текущая реализация системы.

1. Flask [2]: Это микрофреймворк, предназначенный для создания веб-приложений на языке программирования Python. В контексте проекта Miminet Flask был выбран в качестве основного фреймворка благодаря своей универсальности и лёгкости в освоении.

2. Flask-Admink [1]: Это расширение микрофреймворка Flask, позволяющее создавать интерфейс для административной панели приложения поверх существующих моделей данных. Среди достоинств Flask-Admin важно отметить простоту и гибкость в построении сложного интерфейса, что существенно ускоряет создание админ-панели, удовлетворяющей всем требованиям проекта. С помощью этого расширения был реализован редактор тестов для тренажёра Miminet.

3. SQLite [5]: В качестве СУБД в Miminet используется SQLite — лёгковесная и встраиваемая база данных. Она используется для хранения таких сущностей, связанных с тренажёром Miminet, как тесты, разделы тестов, вопросы и категории вопросов.

4. SQLAlchemy [4]: Это библиотека для работы с базами данных на языке Python, которая используется в веб-приложениях для взаимодействия с реляционными СУБД. Её ключевой особенностью является поддержка объектно-реляционного отображения (ORM), что позволяет разработчикам работать с данными в виде Python-объектов, абстрагируясь от низкоуровневых SQL-запросов.

Таким образом, задача выполнялась с помощью уже задействованного в проекте стека технологий, что позволило минимизировать риски

и упростить процесс разработки.

3. Описание решения

Генерация списка вопросов для раздела будет осуществляться по заданной в редакторе разделов строке в формате JSON. Строка будет состоять из ключа — категории вопроса и значения — количества вопросов из этой категории.

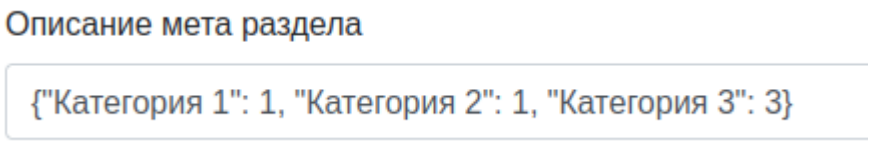

Рис. 1: Пример правильно заданной строки

Строка, полученная из формы админ-панели, будет храниться в новом поле таблицы Section — meta_description. Если строка пустая, значит для раздела не требуется генерация списка случайных вопросов и формирование раздела происходит как обычно; в противном случае на основе полученных данных генерируется список.

3.1. Расширение интерфейса админ-панели

Для задания формы ввода данных в классе SectionView, отвечающем за представление страницы с редактированием раздела теста, Flask-Admin берёт все поля модели за исключением первичного ключа id и полей, указанных в атрибуте form_excluded_columns в базовом классе MiminetAdminModelView, и автоматически преобразует их в соответствующие WTForms-поля [6]. Таким образом, для столбца meta_description будет сформировано поле StringField для ввода строки в формате JSON. Также для отображения столбца в таблице списка записей были дополнены атрибуты column_list и column_labels.

3.2. Генерация списка случайных вопросов

Создание сессии прохождения раздела теста происходит в функции *start_session(section_id: str, user: User)*. Здесь в таблицу QuizSession,

предназначенную для хранения текущих сессий прохождения разделов, добавляются id раздела и пользователя. После этого отсортированные по id раздела вопросы последовательно добавляются в таблицу SessionQuestion, которая содержит информацию о вопросах раздела, вместе с идентификатором текущей сессии. Для добавления автоматической генерации списка произвольных вопросов поле meta_description проверяется на пустоту; затем, если оно не пустое, содержащаяся в нём строка считывается с помощью пакета json [7]. На основе полученных данных создаётся список вопросов заданного размера, упорядоченный по категориям, который впоследствии перемешивается.

3.3. Исправление отображения количества вопросов на странице с разделами

Количество вопросов раздела для отображения на странице теста задаётся в функции *to_section_dto_list(sections: List[Section])* через переменную question_count. Это количество соответствует числу вопросов, связанных с разделом через внешний ключ section_id. Для разделов с заполненным полем meta_description, где вопросы могут ссылаться на другие разделы, был реализован вспомогательный метод *count_meta_section_questions(section_id: str)*, обеспечивающий точный подсчёт количества вопросов.

Заключение

В процессе работы были выполнены следующие задачи:

- дополнен интерфейс редактора разделов в админ-панели;
- реализована генерация списка случайных вопросов для формирования раздела теста;
- исправлено отображение количества вопросов для разделов с непустым полем meta_description;
- проведена апробация в команде;
- исходный код решения доступен в GitHub-репозитории[1];

[1] https://github.com/mimi-net/miminet/tree/meta-section-implementation

Список литературы

[1] Flask-Admin documentation. — URL: https://flask-admin.readthedocs.io/en/stable/.

[2] Flask documentation. — URL: https://flask.palletsprojects.com/en/stable/.

[3] Miminet. — URL: https://miminet.ru/.

[4] SQLAlchemy whith Flask documentation. — URL: https://flask-sqlalchemy.readthedocs.io/en/stable/.

[5] SQLite documentation. — URL: https://www.sqlite.org/docs.html.

[6] WTForms documentation. — URL: https://wtforms.readthedocs.io/en/3.2.x/.

[7] json package documentation. — URL: https://docs.python.org/3/library/json.html.