Информация о работе с кейсами

В Проектно-исследовательской работе кейс — это учебная задача, которую команда должна решить, используя свои навыки в разработке.

Далее вы найдете описание кейсов. Ваши действия:

1. Ознакомиться с основными этапами работы над кейсом. Обратите внимание: этапы одинаковые для всей кейсов.
2. Ознакомиться с кейсами.
3. В команде совместно решить, какой кейс вы будете решать.
4. Зарегистрировать свою команду. Так вы пройдете пункт 1 оценивания. Следите за дедлайном!

Этапы работы над кейсом

1. Pycharm. Отладчик

Создайте новый Pycharm-проект. Реализуйте требуемую в вашей задаче логику, декомпозировав её в классы. Обратите внимание: серверную обвязку делать не нужно.

Создайте main.py и для демонстрации работы реализованной логики напишите примеры использования ваших классов. Пройдитесь отладчиком по написанному коду.

2. Pytest. Coverage

Подключите Pytest к вашему проекту и покройте классы с логикой тестами на 100% по бранчам.

3. Terminal. Git basics

Создайте локальный git-репозиторий и добавьте файлы вашего проекта в него. Настройте .gitignore.

4. GitHub. Git branches

Создайте публичный GitHub-репозиторий. Загрузите ваш локальный проект в него.

Создайте README-файл и заполните в нём описание вашего проекта и заметки к реализации.

5. CI. PR. Git hooks. Linter

Добавьте к вашему проекту CI, который бы запускал тесты, проверял покрытие классов с логикой и соблюдение PEP 8.

6. HTTP и requests

Отведите ветку apispec. В ней отредактируйте ваш README.md-файл, расписав эндпойнты вашего будущего RESTful-сервера. Создайте, но не сливайте pull request к основной ветке.

Пришлите pull request на ревью преподавателю. После согласования и одобрения представленной схемы закройте pull request, влив изменения в основную ветку.

7. API

Отведите ветку server. Напишите тесты на ваш будущий сервер, используя библиотеку requests. Запушьте изменения в созданную ветку. Обратите внимание: CI не должен проходить, потому что сервера ещё не существует. Пишите тесты так, будто сервер уже существует и запущен по адресу: http://localhost:8008

8. HTTP-server. Postman

Реализуйте ваш HTTP-сервер. Отредактируйте тесты, чтобы они тестировали ваш сервер.

9. MongoDB. Pymongo (этап повышенной сложности)

Перенесите хранение данных из памяти в MongoDB. При этом сделайте так, чтобы при тестировании MongoDB мокировалась через mongomock.

Описание кейсов

Кейс 1. Список дел

Сложность: обычная

Вам предстоит разработать сервис списка дел.
Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- добавляет новую задачу в список;
- отмечает, какие задачи уже выполнены;
- показывает весь список задач;
- показывает невыполненные задачи.

Кейс 2. Конвертер валют

Сложность: обычная

Вам предстоит разработать сервис конвертера валют.
Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- пересчитывает сумму из одной валюты в другую по запросу;
- присылает список всех операций, которые были запрошены.

Сервер должен считывать курсы валют из csv-файла.

Кейс 3. Трекер трат

Сложность: обычная

Вам предстоит разработать сервис трекера трат.
Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- добавляет трату (формате «текст») в категорию (формате «текст»), пример: трату «Молоко» в категорию «Еда»; в запросе также должна быть указана потраченная сумма и дата в формате «день.месяц»;

- показывает категорию, на которую пришлось больше всего трат за указанный месяц;
- показывает самую крупную трату за указанный месяц и в указанной категории.

Кейс 4. Трекер диеты

Сложность: обычная

Вам предстоит разработать сервис трекера диеты.
Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- устанавливает цель по калориям на день;
- записывает употреблённые продукты, с указанием названия продукта, калорийности (в расчете на 1 штуку или 100 грамм), количество употреблённых единиц продукта и порядковый номер дня диеты;
- подсчитывает, сколько дней подряд удалось придерживаться режима: максимальная и последняя серии.

Кейс 5. Подбор рецептов

Сложность: обычная

Вам предстоит разработать сервис подбора рецептов.
Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- добавляет рецепт: указывается название, рейтинг и список продуктов с нужным количеством для приготовления;
- подбирает рецепт: пользователь вводит продукты и их количество — в ответ получает список подходящих рецептов, отсортированных по рейтингу.

Кейс 6. Крестики-нолики

Сложность: обычная

Вам предстоит разработать сервис стандартной игры «Крестики-нолики».

Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- создаёт новую игру;
- принимает ходы в игре от имени пользователя (указывается игра и координаты), в ответ отправляется ход программы. Логика автоматического хода программой должна быть следующая: если одним ходом можно выиграть, то программа выигрывает, иначе ходит случайно.

Кейс 7. 5 букв

Сложность: обычная

Вам предстоит разработать сервис стандартной игры в слова (5 букв, Wordle).

Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- создаёт новую игру (для неё слово выбирается случайно из словаря);
- реализует попытку угадать слово с указанием конкретной игры; ответ сервера содержит информацию, согласно правилам игры: какие буквы угаданы, какие из них на своих местах. Если попыток больше нет, должен возвращаться соответствующий код ошибки в запросе.

Кейс 8. Виртуальный питомец

Сложность: повышенная

Вам предстоит разработать сервис виртуального питомца.

Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- создаёт питомца с указанием показателя жизни (число) и клички;

- даёт возможность совершать действия с питомцем: покормить, погладить, дать вкусняшку и погулять;
- показывает текущую статистику по питомцу.

Питомец имеет:

- имя — изначально берётся из запроса при создании;
- показатель жизни — изначально берётся из запроса при создании;
- показатель счастья — изначально 2;
- показатель голода — изначально 2.

С питомцем можно совершить следующие действия:

- покормить (-2 к голоду);
- погладить (+1 к счастью, +1 к голоду);
- дать вкусняшку (+2 к счастью, -1 к голоду);
- погулять (+1 к счастью, +3 к голоду);
- игнорировать (-1 к счастью, +1 к голоду).

Если после какого-либо действия голод превышает 5, то у питомца отнимается 1 очко жизни. Если после предыдущего действия уже уменьшалась жизнь, то отнимается на 1 очко больше, чем отнималось в прошлый раз.

Если показатель жизни достигает 0, то питомец исчезает. Если после какого-либо действия счастье питомца достигает 0, то питомец исчезает.

Кейс 9. Рекомендательная система

Сложность: повышенная, требует математики

Вам предстоит разработать сервис рекомендательной системы.

Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- создает нового пользователя;
- добавляет информацию like / dislike;
- показывает список рекомендованых непросмотренных позиций.

Предлагаемый алгоритм рекомендаций:

1. Отсортировать для конкретного пользователя всех других пользователей по похожести предпочтений.
2. Пройтись по пользователям в полученном порядке и извлечь непросмотренные исходным пользователем отмеченные лайком фильмы.

Похожесть одного пользователя на другого определяется по косинусной метрике. Пример.

Кейс 10. Поисковик

Сложность: повышенная+, требует математики

Вам предстоит разработать сервис поиска по текстовым pdf-файлам.

Это должен быть HTTP-сервер с RESTful API, который принимает запросы и умеет выполнять следующие операции:

- в ответ на текстовый запрос показывает ранжированный результат поиска из максимум 10 самых релевантных документов.

Требования к сервису поиска по текстовым pdf-файлам:

- запрос может состоять из нескольких слов;
- в yml-файле настроек сервера указываются папки, по которым осуществляется поиск;
- при старте сервер проверяет наличие новых файлов и если надо доиндексирует их;
- ранжирование осуществляется на основе tf-idf модели;
- весь текст рассматривается без учёта регистра.

Другую функциональность реализовывать не нужно, например, не надо стемматизировать или лемматизировать слова. Для чтения pdf используйте модуль pypdf.