Antida Python School

Техническое задание на разработку ПО

«Сервис для учёта личных финансов»
Серверная часть

2020

1 Краткое описание предметной области

Разрабатываемое ПО представляет собой серверную часть для сервиса учёта финансов. Сервис позволяет пользователям фиксировать свои доходы и расходы, а также получать личные финансовые отчёты.

2 Требования к технической части

2.1 Применяемый стек технологий

- Язык программирования — Python версии 3.8+
- Веб-фреймворк — Flask версии 1.1.2+
- Система управления базами данных — SQLite версии 3.31+

2.2 Требования к оформлению программного кода

- Соблюдение рекомендаций PEP8
- Документирование функций, классов, методов (Docstring)

2.3 Требования к инфраструктуре проекта

- Применение системы контроля версий Git
- Наличие в репозитории проекта следующих файлов:
  - readme.md с описанием проекта и инструкцией по развёртыванию
  - example.db с БД, заполненной тестовыми данными
  - физическая модель БД в виде изображения
  - коллекция Postman-запросов для тестирования, покрывающих все эндпоинты
- Зависимости должны быть зафиксированы. Способ фиксации — один из ниже перечисленных, на усмотрение разработчика:
  - файл requirements.txt
  - библиотека pipenv
  - библиотека poetry (наиболее предпочтительный вариант)

3 Функциональные требования

Разрабатываемое ПО должно обеспечивать реализацию REST API для выполнения ниже перечисленных функций.

Общие примечания к описанным функциям:

- Запросы и ответы представлены в JSON-подобной структуре и описывают JSON-объекты
- Типы полей указываются через двоеточие
- Поля, помеченные знаком «?» – не обязательные
- Запись вида [type] обозначает список значений типа type
- Поле “date” во всех запросах и ответах передаётся в стандарте UTC, формат UTS
- Поле “date” во всех запросах и ответах может содержать не наступившую дату/время
- Поле “type” во всех запросах имеет логический тип. Для операций прихода его значение ”true”, для расходных операций – ”false”
- Значение поля “amount” передаётся в виде строки, разделитель дробной части – точка. В случае передачи более двух знаков после запятой – должно происходить округление до двух знаков.

3.1 Регистрация пользователя

Структура запроса:

POST /register
REQUEST:
{
“first_name”: str,
“last_name”: str,
“email”: str,
“password”: str
}
RESPONSE:
{
“id”: int,
“first_name”: str,
“last_name”: str,
“email”: str
}

Описание полей запроса/ответа:

Поле	Назначение
first_name	Имя пользователя
last_name	Фамилия пользователя
email	Адрес электронной почты пользователя
password	Пароль для создаваемого аккаунта

Логика работы:

- Хранение паролей в открытом виде запрещено. Принимаемый пароль должен быть сохранён в виде хеша. Алгоритм хеширования — на усмотрение разработчика
- Поле “email” используется в качестве логина и является уникальным в пределах разрабатываемого сервиса

3.2 Авторизация. Вход в учётную запись

Структура запроса:

POST /auth/login
Request:
{
"email": str,
"password": str
}

Описание полей запроса/ответа:

Поле	Назначение
email	Логин (адрес электронной почты пользователя)
password	Пароль

3.3 Авторизация. Выход из учётной записи

Структура запроса:

POST /auth/logout

3.4 Категория. Создание

Структура запроса:

POST /category
REQUEST:
{
“name”: str,
“parent_id”: int?
}
RESPONSE:
{
“id”: int,
“name”: str,
“parent_id”: int?
}

Описание полей запроса/ответа:

Поле	Назначение
id	Идентификатор категории
name	Имя категории
parent_id	Идентификатор родительской категории

Логика работы:

- Метод доступен только авторизованным пользователям
- Каждый пользователь имеет своё дерево категорий. Категория привязывается к пользователю, создавшему её и помещается в его дерево категорий
- Имена категорий и подкатегорий уникальны и не могут повторяться в пределах дерева одного пользователя (даже если имеют разные родительские категории)
- Если родительская категория не указана — создаётся категория верхнего уровня
- Если категория уже существует — формируется ответ с её параметрами
- Глубина вложенности в дереве категорий — не ограничена

3.5 Категория. Получение

Структура запроса:

GET /category
REQUEST:
{
“name”: str,
}
RESPONSE:
{
“id”: int,
“name”: str,
“parent_id”: int?
}

Описание полей запроса/ответа:

Поле	Назначение
id	Идентификатор категории
name	Имя категории
parent_id	Идентификатор родительской категории

Логика работы:

- Метод доступен только авторизованным пользователям
- Поиск производится по уникальному имени категории
- Поиск производится в дереве того пользователя, который выполняет запрос

3.6 Категория. Редактирование

Структура запроса:

PATCH /category/<id>
REQUEST:
{
“name”: str?,
“parent_id”: int?
}
RESPONSE:
{
“id”: int,
“name”: str,
“parent_id”: int?
}

Описание полей запроса/ответа:

Поле	Назначение
id	Идентификатор категории
name	Имя категории
parent_id	Идентификатор родительской категории

Логика работы:

- Метод доступен только авторизованным пользователям
- Пользователь может редактировать только созданные им категории
- Если поле “parent_id” присутствует в запросе, но содержит значение “undefined” – категория преобразуется в категорию верхнего уровня

3.7 Категория. Удаление

Структура запроса:

DELETE /category/<id>

Логика работы:

- Метод доступен только авторизованным пользователям
- Пользователь может удалять только созданные им категории
- При удалении категории все привязанные к ней операции становятся безкатегорийными, а потомки ближайшего к удаляемой категории уровня — становится категориями верхнего уровня, сохраняя свои деревья.

3.8 Операции. Создание

Структура запроса:

POST /transactions
REQUEST:
{
“type”: bool,
“amount”: str,
“description”: str?,
“category_id”: int?,
“date”: int?
}
RESPONSE:
{
“id”: int
“type”: bool,
“amount”: str,
“description”: str?,
“category_id”: int?,
“date”: int
}

Описание полей запроса/ответа:

Поле	Назначение
type	Тип операции
amount	Сумма операции
description	Описание операции
category_id	Идентификатор категории
date	Дата и время
id	Идентификатор операции

Логика работы:

- Метод доступен только авторизованным пользователям
- Операция может иметь не более 1 категории
- Операция может не иметь категории
- Если поле “date” не передано — применяется время сервера (UTC)

3.9 Операции. Редактирование

Структура запроса:

PATCH /transactions/<id>
REQUEST:
{
“type”: bool?,
“amount”: str?,
“description”: str?,
“category_id”: int?,
“date”: int?
}
RESPONSE:
{
“id”: int
“type”: bool,
“amount”: str,
“description”: str?,
“category_id”: int?,
“date”: int
}

Описание полей запроса/ответа:

Поле	Назначение
type	Тип операции
amount	Сумма операции
description	Описание операции
category_id	Идентификатор категории
date	Дата и время
id	Идентификатор операции

Логика работы:

- Метод доступен только авторизованным пользователям
- Метод доступен только для операций, которые созданы пользователем, выполняющим запрос
- Операция может иметь не более 1 категории

3.10 Операции. Удаление

Структура запроса:

DELETE /transactions/<id>

Логика работы:

- Метод доступен только авторизованным пользователям
- Метод доступен только для операций, которые созданы пользователем, выполняющим запрос

3.11 Операции. Получение отчёта

Структура запроса:

```
GET /transactions
Query string:
    “category_id”:          int?,
    “from”:                 int?,
    “to”:                   int?,
    “period”:               str?,
    “page_size”:            int?,
    “page”:                 int?
RESPONSE:
{
    “operations”: [
        {
            “id”:               int,
            “date”:             int,
            “type”:             bool,
            “description”:      str?,
            “amount”:           str,
            “categories”:       [
                {
                    “id”:       int,
                    “name”:     str,
                }

            ]
        }
    ],
    “total”:                int,
    “total_items”:          int,
    “total_pages”:          int,
    “page_size”:            int,
    “page”:                 int,
    “next_page”:            str?,
    “prev_page”:            str?
}
```

Описание query string параметров:

Поле	Назначение
category_id	Идентификатор категории
from	Начальная граница диапазона временного периода
to	Конечная граница диапазона временного периода
period	Предустановленный временной период
page	Страница отчёта
page_size	Максимальное количество элементов на странице

Виды предустановленных периодов (поле ”period”):

- week – текущая неделя с понедельника по воскресенье
- last_week – предыдущая неделя с понедельника по воскресенье
- month – текущий месяц
- last_month – предыдущий месяц
- quarter – текущий квартал
- last_quarter – предыдущий квартал
- year – текущий год
- last_year – предыдущий год

Описание полей запроса/ответа:

Поле	Назначение
operations	Массив объектов “Операция”
id	Идентификатор операции
date	Дата и время
type	Тип операции
description	Описание операции
amount	Сумма операции
categories	Массив объектов “Категории”. Хранит весь путь для категорий от категории текущей операции до её корневой категории.
id (categories)	Идентификатор категории
name	Имя категории
total	Сумма по всему отчёту
total_items	Количество элементов в отчёте
total_pages	Количество страниц в отчёте
page_size	Максимальное количество элементов на странице
page	Текущая страница отчёта
next_page	Ссылка на получение следующей страницы отчёта
prev_page	Ссылка на получение предыдущей страницы отчёта

Логика работы:

- Метод доступен только авторизованным пользователям.
- Метод доступен только для операций, которые созданы пользователем, выполняющим запрос.
- Все query-string параметры не обязательны.
- Все query-string параметры могут иметь только одно значение.
- Если запрос выполнен без параметров — в отчёт включается вся накопленная история операций пользователя по всем категориям и за всё время
- Максимальное количество записей по умолчанию на одной странице (если поле “page_size” не передано) — 20 записей.
- Номер текущей страницы отчёта по умолчанию (если поле “page” не передано) — первая страница.
- Если передана категория (поле ”category”) — выполняется фильтрация по категориям. Отчёт строится для категории и всех её подкатегорий на всю глубину вложенности.
- Если передано начало временного периода (поле ”from”) — выполняется фильтрация по дате операции. В отчёт включаются только те записи, дата которых превышает заданную.
- Если передано окончание временного периода (поле ”to”) — выполняется фильтрация по дате операции. В отчёт включаются только те записи, дата которых не превышает заданную.
- Если передан один из видов предустановленных периодов (поле ”period”) — фильтрация по дате осуществляется для выбранного периода.
- Фильтр по предустановленному периоду имеет приоритет выше, чем фильтры по началу и окончанию периода. Фильтры по началу/окончанию периода игнорируются в случае, если они переданы совместно с фильтром по предустановленному периоду.
- Если передано максимальное количество записей на странице (поле “page_size”), то используется указанный размер страниц.
- Все описанные фильтры могут применяться одновременно и не оказывают взаимного влияния друг на друга, за исключением оговоренных случаев.
- Записи в отчёте сортируются по дате.
- Объекты “Категория” в массиве поля “categories” расположены в порядке от категории верхнего уровня к категориям нижних уровней.
- При работе с БД принимается допущение, что данные по операциям текущего пользователя не могут быть изменены с другого клиента. В таком случае при постраничном выводе каждой новой страницы отчёта допустимо делать новый запрос к БД, а не хранить полный «слепок» последнего запроса для обеспечения его достоверности.

4 Порядок сдачи результатов

4.1 Общие положения

4.1.1 Качество выполнения функций проверяется согласно техническому заданию.
4.1.2 В репозитории должен присутствовать readme-файл с описанием проекта и его эксплуатационной документацией.
4.1.3 Приемка подсистемы должна включать проверку результатов тестирования.
4.1.4 Тесты разрабатываются в виде коллекции postman-запросов.
4.1.5 Тесты должны покрывать весь функционал программы.
4.1.6 Тесты разрабатывает команда разработчиков.
4.1.7 После завершения приёмки исходный код передаётся заказчику

4.2 Этапы разработки

4.2.1 Этап 1. Согласование общей концепции

Продолжительность этапа: 7 дней

На первом этапе составляются и согласуется с заказчиком следующие документы:

- настоящее ТЗ
- концептуальная модель БД
- логическая модель БД
- физическая модель БД

4.2.2 Этап 2. Минимально жизнеспособный продукт (MVP)

Продолжительность этапа: 7 дней

На данном этапе команда разрабатывает и демонстрирует заказчику MVP. Разработанный продукт может содержать не все оговоренные в настоящем ТЗ функции и/или не в полном их объёме. Глубина проработки выбирается на усмотрение команды.

4.2.3 Этап 3. Завершение разработки

Продолжительность этапа: 7 дней

Это завершающий этап, по окончании которого осуществляется приёмка разработанного ПО согласно требованиям настоящего ТЗ и пункта 4.1 «Общие положения».