**Чек-лист API тестов**

Корректность структуры данных

**POST запросы**

* Заполнены все поля валидными данными
* Заполнены только обязательные поля
* Заполнены не все обязательные поля
* Не заполнено ни одно поле
* Валидация данных в полях (корректные и некорректные данные)
* Пустой JSON
* Дата создания объекта

**GET запросы**

* Пустой список (если возможно)
* Заполненный список
* Пагинация в списке (limit, offset)
* Получение списка с ограничением на количество записей
* Получение списка начиная с указанного номера
* В случае передачи параметров с некорректным значением возвращается 400 с описанием ошибки в теле ответа
* При отрицательном offset список пользователей возвращается, начиная с первой позиции
* При несуществующем offset возвращается пустой список пользователей
* Сортировка списка
* Запрос данных по валидному ID, проверка возвращения корректных данных
* Запрос данных по несуществующему ID, но в валидном формате
* Запрос данных по невалидному ID

**PUT запросы**

* Обновление с корректными данными
* Обновление по несуществующему ID
* Обновление по невалидному ID
* Валидация полей (корректные и некорректные данные)
* Частичное обновление (в JSON присутствуют не все поля)

**DELETE запросы**

* Удаление существующего объекта
* Удаление уже удаленного объекта
* Удаление по несуществующему ID
* Удаление по невалидному ID
* Удаление и повторное добавление такой же сущности (если есть уникальные поля)

**Проверка статусов ответов**

**Проверка всех возможных ошибок**

**Остальные специфичные проверки в случае сложной логики**

**Тестовые идеи для проверки API.**

* Отправить (GET, POST, PUT, DELETE, PATCH) с обязательными валидными параметрами или валидным телом в запросе . Проверить ответ от сервера согласно требованиям.
* проверить версию протокола
* проверить код состояния согласно ТЗ, например, что запрос должен возвращать код состояния HTTP 2XX для GET, POST, DELETE, PUT
* проверить пояснение (status text)
* проверить сформированный объект JSON, а именно структуру ответа согласно модели данных из требований
* проверить объект в формате JSON с запрошенными полями
* проверить имена свойств (ключей), типы полей, значения полей согласно требованиям
* проверить, что у полей значения не NULL в полученном ответе
* проверить, что у полей значения не являются пустыми
* проверить состояние после отправки запросов, например, для запросов GET убедитесь, что система не изменила состояние, проверить идемпотентность
* проверить идемпотентность для методов PUT и DELETE
* проверить основные заголовки HTTP headers, заголовки ответа согласно требованиям

проверить, что информация не утекает через заголовки (например, X-Powered-By заголовок не отправляется пользователю)

* проверить создание токена авторизации
* проверить структуру полезных данных (payload) токена, который представлен закодированной строкой
* проверить различные типы токенов авторизации согласно ТЗ
* проверить создание id или uuid
* проверить обновление существующих данных
* проверить удаление существующего объекта
* проверить, что ответ на запрос приходит своевременно
* проверить, что токен авторизации действительный
* проверить, параметры при отправке запроса
* проверить, тело при получении запроса

**Негативные проверки**

Отправить данные с

- невалидным токеном

- пустым токеном

- несуществующим или невалидным id

- неполной JSON моделью с отсутствующими полями

- недействительными значениями,

- невалидным или недопустимым uuid (длиной порядка 500 символов или больше)

- пустыми или невалидными HTTP заголовками

- неподдерживаемые методами,

- некорректным телом в запросе

- неверным типом параметра

- недопустимыми символами в параметрах или в теле запроса

- переполнением тела ответа запроса при отправке (большой JSON)

- пустыми значениями во вложенных объектах, массивах в формате JSON

- полностью пустым JSON

- с нулевым значением поля

- с дубликатами значений в теле запроса

- с комментариями в теле запроса

Ожидаемый результат при негативном тестировании это должны быть корректный статус, корректное сообщение, и корректное тело ответа от сервера согласно ТЗ

Дополнительные проверки:

* проверить ответ на успешный запрос согласно ТЗ
* проверить ответ на отклоненный запрос согласно ТЗ
* перехватить запрос при отправке, заменить в url параметры, проверить ответ от сервера
* перехватить ответ от сервера, заменить значения поля, проверить отображение данных полей в системе
* при перехвате запроса, заменить метод и потом проверить ответ от сервера
* при перехвате ответа от сервера, заменить тело запроса, проверить реакцию системы и отображение соответствующих данных
* при перехвате запроса, заменить тело запроса и потом проверить ответ от сервера
* создать ресурс с уже существующим именем и проверить ответ от сервера
* удалить параметр в url при перехвате запроса на сервер и проверить ответ от сервера
* удалить поле со значением при перехвате ответа от сервера, проверить реакцию системы
* удалить тело запроса при перехвате запроса, проверить ответ от сервера
* удалить ресурс, который не существует, уже удалён
* удалить значение поля по невалидному id
* обновить ресурс с недопустимыми данными, несуществующими данными
* обновить значение поля для несуществующего или невалидного id
* проверить, что код состояния HTTP соответствует типу ошибки согласно ТЗ
* проверить частичное обновление данных
* проверить коды ошибок и их сообщения согласно ТЗ
* проверить формат ошибки согласно ТЗ (есть сообщения об ошибке, которые не должны отображаться в системе, но должны отображаться в логах)
* отправить одновременно несколько запросов, проверить ответ от сервера
* проверить границы параметров/значений полей согласно ТЗ при отправке запросов
* проверить с запросы с заполненными необязательными полями
* проверить дату создания объекта при наличии этого поля (соответствия времени с сервером или разница с часовыми поясами)
* проверить ответы от сервера в системных и интеграционных журналах, в логах
* проверить дополнительные параметры эндпоинта: фильтрацию, сортировку, лимитирование, пропуск, список
* проверить безопасность и кэшируемость методов