Отчет о тестировании API

1 Информация о тестировании

- Название проекта: travel-planner-backend
- Версия API: v1
- Дата тестирования: 05.05.2025

2 Описание тестирования

Проведено ручное тестирование с использованием Bruno. Тестировалась функциональность эндпоинтов API у сервисов: auth, external, library, planner.

3 Результаты тестирования

3.1 Сервис library

3.1.1 Получение списка публичных маршрутов (GET /routes)

Параметр	Описание
Название теста	Получение списка публичных маршрутов (базовый случай)
Цель	Проверить успешное получение списка публичных маршрутов без фильтров с пагинацией по умолчанию.
Эндпоинт	GET /routes
Предусловия	API доступно. Существуют публичные маршруты в базе данных.
Тип тестирования	Позитивный
Данные запроса	Отсутствуют (используются параметры по умолчанию: page=1, limit=20).
Шаги выполнения	1. Сформировать GET-запрос к /routes. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив data с объектами RoutePreview и объект pagination.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит поля data (массив) и pagination (объект).

	- Структура элементов в data соответствует схеме RoutePreview. - Структура объекта pagination соответствует схеме Pagination.
Полученный результат	Статус код 200. Ответ содержит корректные поля data и pagination. Структура соответствует ожидаемой.

3.1.2 Публикация маршрута в библиотеке (POST /routes)

Параметр	**Описание**
Название теста	Успешная публикация нового маршрута
Цель	Проверить возможность публикации нового маршрута авторизованным пользователем.
Эндпоинт	POST /routes
Предусловия	Пользователь авторизован (валидный Bearer токен). Существует маршрут с указанным routeId в planner-сервисе, доступный пользователю.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Тело запроса (JSON): { "routeId": "...", "title": "...", "description": "...", "tags": [...] } (валидные данные)
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос с валидным телом. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит созданный объект PublicRoute.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект, соответствующий схеме PublicRoute. - Поля title, description, tags, originalRouteId в ответе соответствуют отправленным. - Маршрут действительно появился в списке (GET /routes).

Полученный результат	Статус код 201. Получен корректный объект PublicRoute. Маршрут доступен через GET-запрос.

3.1.3 Получение детальной информации о маршруте (GET /routes/{routeId})

Параметр	**Описание**
Название теста	Получение деталей существующего публичного маршрута
Цель	Проверить успешное получение полной информации о конкретном публичном маршруте по его ID.
Эндпоинт	GET /routes/{routeId}
Предусловия	API доступно. Публичный маршрут с указанным {routeId} существует.
Тип тестирования	Позитивный
Данные запроса	Path parameter: routeId (валидный UUID существующего маршрута).
Шаги выполнения	1. Сформировать GET-запрос к /routes/{routeId}, подставив ID. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект PublicRouteDetail с полной информацией о маршруте.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме PublicRouteDetail. - Поле id в ответе совпадает с запрошенным routeId. - Присутствует массив days с информацией о местах.
Полученный результат	Статус код 200. Получен корректный объект PublicRouteDetail с запрошенным ID и деталями маршрута.

3.1.4 Обновление публичного маршрута (PUT /routes/{routeId})

Параметр	**Описание**

Название теста	Успешное обновление данных публичного маршрута владельцем
Цель	Проверить возможность обновления информации (title, description, tags) публичного маршрута его автором.
Эндпоинт	PUT /routes/{routeId}
Предусловия	Пользователь авторизован (валидный Bearer токен). Маршрут с {routeId} существует, и авторизованный пользователь является его автором.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: routeId (ID маршрута пользователя) Тело запроса (JSON): { "title": "...", "description": "...", "tags": [...] } (новые валидные данные)
Шаги выполнения	1. Авторизоваться как владелец маршрута. 2. Сформировать PUT-запрос с новыми данными в теле. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект PublicRoute.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме PublicRoute. - Поля title, description, tags в ответе содержат обновленные значения. - Поле updatedAt изменилось. - При последующем GET-запросе к этому маршруту возвращаются обновленные данные.
Полученный результат	Статус код 200. Получен объект PublicRoute с обновленными данными. Изменения подтверждены GET-запросом.

3.1.5 Удаление публичного маршрута (DELETE /routes/{routeId})

Параметр	**Описание**
Название теста	Успешное удаление публичного маршрута

	владельцем
Цель	Проверить возможность удаления публичного маршрута его автором.
Эндпоинт	DELETE /routes/{routeId}
Предусловия	Пользователь авторизован (валидный Bearer токен). Маршрут с {routeId} существует, и авторизованный пользователь является его автором.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: routeId (ID маршрута пользователя)
Шаги выполнения	1. Авторизоваться как владелец маршрута. 2. Сформировать DELETE-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - При последующем GET-запросе к этому {routeId} возвращается ошибка 404 (Not Found).
Полученный результат	Статус код 204. Тело ответа пустое. Маршрут недоступен через GET-запрос (возвращается 404).

3.1.6 Получение отзывов о маршруте (GET /routes/{routeId}/reviews)

Параметр	**Описание**
Название теста	Получение списка отзывов для существующего маршрута
Цель	Проверить успешное получение списка отзывов для конкретного публичного маршрута с пагинацией.
Эндпоинт	GET /routes/{routeId}/reviews
Предусловия	API доступно. Публичный маршрут с

	указанным {routeId} существует. Для маршрута существуют отзывы.
Тип тестирования	Позитивный
Данные запроса	Path parameter: routeId (валидный UUID существующего маршрута). Query parameters: page=1, limit=10 (пример).
Шаги выполнения	1. Сформировать GET-запрос к /routes/{routeId}/reviews с параметрами пагинации. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив data с объектами Review и объект pagination.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит поля data (массив) и pagination (объект). - Структура элементов в data соответствует схеме Review. - Поле routeId у каждого отзыва в data совпадает с запрошенным. - Параметры page и limit в pagination соответствуют запрошенным.
Полученный результат	Статус код 200. Ответ содержит корректные поля data (с отзывами) и pagination. Структура соответствует ожидаемой.

3.1.7 Добавление отзыва к маршруту (POST /routes/{routeId}/reviews)

Параметр	**Описание**
Название теста	Успешное добавление нового отзыва к маршруту
Цель	Проверить возможность добавления нового отзыва авторизованным пользователем к существующему маршруту.
Эндпоинт	POST /routes/{routeId}/reviews
Предусловия	Пользователь авторизован (валидный Bearer токен).

	Маршрут с {routeId} существует. Пользователь еще не оставлял отзыв к этому маршруту.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: routeId (ID существующего маршрута) Тело запроса (JSON): { "rating": 4.5, "comment": "..." } (валидные данные)
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос с валидным телом. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит созданный объект Review.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект, соответствующий схеме Review. - Поля rating, comment, routeId и author (ID пользователя) в ответе корректны. - Отзыв действительно появился в списке (GET /routes/{routeId}/reviews). - Обновился reviewsCount и rating у маршрута (GET /routes/{routeId}).
Полученный результат	Статус код 201. Получен корректный объект Review. Отзыв доступен через GET-запрос. Счетчик и рейтинг маршрута обновлены.

3.1.8 Обновление отзыва (PUT /routes/{routeId}/reviews/{reviewId})

Параметр	**Описание**
Название теста	Успешное обновление существующего отзыва его автором
Цель	Проверить возможность обновления rating и comment существующего отзыва его автором.
Эндпоинт	PUT /routes/{routeId}/reviews/{reviewId}
Предусловия	Пользователь авторизован (валидный Bearer токен). Маршрут с {routeId} существует. Отзыв

	с {reviewId} существует, и авторизованный пользователь является его автором.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: routeId, reviewId (ID отзыва пользователя) Тело запроса (JSON): { "rating": 5, "comment": "..." } (новые валидные данные)
Шаги выполнения	1. Авторизоваться как автор отзыва. 2. Сформировать PUT-запрос с новыми данными в теле. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект Review.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме Review. - Поля rating, comment в ответе содержат обновленные значения. - Поле updatedAt изменилось. - При последующем GET-запросе к списку отзывов возвращаются обновленные данные для этого отзыва. - Обновился rating у маршрута (GET /routes/{routeId}).
Полученный результат	Статус код 200. Получен объект Review с обновленными данными. Изменения подтверждены GET-запросом. Рейтинг маршрута обновлен.

3.1.9 Удаление отзыва (DELETE /routes/{routeId}/reviews/{reviewId})

Параметр	**Описание**
Название теста	Успешное удаление отзыва его автором
Цель	Проверить возможность удаления отзыва его автором.
Эндпоинт	DELETE /routes/{routeId}/reviews/{reviewId}

Предусловия	Пользователь авторизован (валидный Bearer токен). Маршрут с {routeId} существует. Отзыв с {reviewId} существует, и авторизованный пользователь является его автором.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: routeId, reviewId (ID отзыва пользователя)
Шаги выполнения	1. Авторизоваться как автор отзыва. 2. Сформировать DELETE-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - При последующем GET-запросе к списку отзывов этот {reviewId} отсутствует. - Обновился reviewsCount и rating у маршрута (GET /routes/{routeId}).
Полученный результат	Статус код 204. Тело ответа пустое. Отзыв отсутствует в списке. Счетчик и рейтинг маршрута обновлены.

3.1.10 Копирование публичного маршрута в личные (POST /routes/{routeId}/copy)

Параметр	**Описание**
Название теста	Успешное копирование публичного маршрута в личные
Цель	Проверить возможность копирования существующего публичного маршрута в личные маршруты авторизованного пользователя.
Эндпоинт	POST /routes/{routeId}/copy
Предусловия	Пользователь авторизован (валидный Bearer токен). Публичный маршрут с {routeId} существует.
Тип	Позитивный

тестирования	
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: routeId (ID существующего публичного маршрута)
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит объект с id нового личного маршрута и сообщением об успехе.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит поля id (UUID) и message (строка). - Новый личный маршрут с полученным id доступен пользователю (через API личных маршрутов, если оно есть).
Полученный результат	Статус код 201. Получен объект с ID нового маршрута и сообщением. Копирование подтверждено (гипотетически, через другое API).

3.1.11 Получение данных для карты маршрута (GET /routes/{routeId}/map)

Параметр	**Описание**
Название теста	Получение данных для карты существующего публичного маршрута
Цель	Проверить успешное получение данных (точки, порядок, цвета) для отображения конкретного публичного маршрута на карте.
Эндпоинт	GET /routes/{routeId}/map
Предусловия	API доступно. Публичный маршрут с указанным {routeId} существует и содержит информацию о днях и местах.
Тип тестирования	Позитивный
Данные	Path parameter: routeId (валидный UUID существующего

запроса	маршрута).
Шаги выполнения	1. Сформировать GET-запрос к /routes/{routeId}/map. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект RouteMapData с информацией для карты.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме RouteMapData. - Поле routeId в ответе совпадает с запрошенным. - Присутствует массив days, каждый элемент которого содержит day, color и массив places. - Каждый элемент places содержит id, name, category, location, order.
Полученный результат	Статус код 200. Получен корректный объект RouteMapData с необходимой структурой для отображения карты.

3.1.12 Получение статуса обновлений маршрута (GET /routes/shared/{routeId}/updates)

Параметр	**Описание**
Название теста	Проверка наличия обновлений в совместно редактируемом маршруте
Цель	Проверить успешное получение статуса обновлений (наличие, дата, автор) для совместно редактируемого маршрута, к которому пользователь имеет доступ.
Эндпоинт	GET /routes/shared/{routeId}/updates
Предусловия	Пользователь авторизован (валидный Bearer токен). Маршрут с {routeId} существует и является совместно редактируемым. Пользователь имеет доступ к этому маршруту.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: routeId (ID совместно редактируемого

	маршрута) Query parameter: lastSyncTimestamp (валидная дата-время)
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос с указанием lastSyncTimestamp. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с полями hasUpdates (boolean), lastModified (date-time), modifiedBy (string, опционально).
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит поля hasUpdates, lastModified. - Типы данных полей соответствуют спецификации. - Значение hasUpdates корректно отражает, была ли модификация после lastSyncTimestamp.
Полученный результат	Статус код 200. Получен корректный объект статуса обновлений. Поле hasUpdates соответствует ожидаемому состоянию.

3.1.13 Подписка на уведомления об изменениях (PUT /routes/shared/{routeId}/notifications/subscribe)

Параметр	**Описание**
Название теста	Успешная подписка на уведомления об изменениях маршрута
Цель	Проверить возможность подписки авторизованного пользователя на push-уведомления об изменениях в совместно редактируемом маршруте.
Эндпоинт	PUT /routes/shared/{routeId}/notifications/subscribe
Предусловия	Пользователь авторизован (валидный Bearer токен). Маршрут с {routeId} существует и является совместно редактируемым. Пользователь имеет доступ к этому маршруту. Пользователь еще не подписан с этим deviceToken.
Тип тестирования	Позитивный

Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: routeId (ID совместно редактируемого маршрута) Тело запроса (JSON): { "deviceToken": "..." } (валидный токен устройства)
Шаги выполнения	1. Авторизоваться. 2. Сформировать PUT-запрос с deviceToken в теле. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с сообщением об успешной подписке.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит поле message с текстом об успехе. - В системе (БД) создана запись о подписке пользователя на уведомления для данного маршрута и устройства.
Полученный результат	Статус код 200. Получено сообщение об успехе. Подписка зарегистрирована в системе.

3.1.14 Отписка от уведомлений об изменениях (PUT /routes/shared/{routeId}/notifications/unsubscribe)

Параметр	**Описание**
Название теста	Успешная отписка от уведомлений об изменениях маршрута
Цель	Проверить возможность отписки авторизованного пользователя от push-уведомлений об изменениях в совместно редактируемом маршруте.
Эндпоинт	PUT /routes/shared/{routeId}/notifications/unsubscribe
Предусловия	Пользователь авторизован (валидный Bearer токен). Маршрут с {routeId} существует. Существует активная подписка для пользователя, данного маршрута и указанного deviceToken.
Тип тестирования	Позитивный
Данные	Заголовок: Authorization: Bearer <token>

запроса	Path parameter: routeId (ID совместно редактируемого маршрута) Тело запроса (JSON): { "deviceToken": "..." } (токен, на который была подписка)
Шаги выполнения	1. Авторизоваться. 2. Сформировать PUT-запрос с deviceToken в теле. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с сообщением об успешной отписке.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит поле message с текстом об успехе. - В системе (БД) удалена запись о подписке.
Полученный результат	Статус код 200. Получено сообщение об успехе. Подписка удалена из системы.

3.2 Сервис planner

3.2.1 Получение всех поездок пользователя (GET /trips)

Параметр	**Описание**
Название теста	Получение списка поездок пользователя (с пагинацией)
Цель	Проверить успешное получение списка поездок авторизованного пользователя с использованием параметров пагинации и фильтрации по статусу.
Эндпоинт	GET /trips
Предусловия	Пользователь авторизован (валидный Bearer токен). У пользователя существуют поездки в системе.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Query parameters: status=upcoming, limit=10, offset=0 (пример).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips с параметрами фильтрации и пагинации.

	3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с total (общее кол-во) и trips (массив объектов TripSummary).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит поля total (число) и trips (массив). - Структура элементов в trips соответствует схеме TripSummary. - Количество элементов в trips не превышает limit. - Если применялся фильтр status, все поездки в ответе имеют этот статус.
Полученный результат	Статус код 200. Ответ содержит корректные поля total и trips. Структура и фильтрация соответствуют ожидаемой.

3.2.2 Создание новой поездки (POST /trips)

Параметр	**Описание**
Название теста	Успешное создание новой поездки
Цель	Проверить возможность создания новой поездки авторизованным пользователем.
Эндпоинт	POST /trips
Предусловия	Пользователь авторизован (валидный Bearer токен).
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Тело запроса (JSON): { "title": "...", "startDate": "...", "endDate": "..." } (валидные данные, соответствующие TripCreate)
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос с валидным телом. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит созданный объект Trip.
Проверка	Проверить, что: - Статус код ответа 201.

	- Тело ответа содержит объект, соответствующий схеме Trip. - Поля title, startDate, endDate в ответе соответствуют отправленным. - Поле creator содержит информацию об авторизованном пользователе. - Поездка действительно появилась в списке (GET /trips).
Полученный результат	Статус код 201. Получен корректный объект Trip. Поездка доступна через GET-запрос.

3.2.3 Получение деталей поездки (GET /trips/{tripId})

Параметр	**Описание**
Название теста	Получение деталей существующей поездки
Цель	Проверить успешное получение полной информации о конкретной поездке по ее ID, к которой пользователь имеет доступ.
Эндпоинт	GET /trips/{tripId}
Предусловия	Пользователь авторизован (валидный Bearer токен). Поездка с указанным {tripId} существует, и пользователь имеет к ней доступ (создатель или через share).
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId (валидный UUID существующей поездки).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips/{tripId}, подставив ID. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект Trip с полной информацией о поездке.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме Trip.

	- Поле id в ответе совпадает с запрошенным tripId.
Полученный результат	Статус код 200. Получен корректный объект Trip с запрошенным ID и деталями поездки.

3.2.4 Обновление деталей поездки (PUT /trips/{tripId})

Параметр	**Описание**
Название теста	Успешное обновление данных поездки пользователем с правом редактирования
Цель	Проверить возможность обновления информации поездки пользователем, имеющим права на редактирование.
Эндпоинт	PUT /trips/{tripId}
Предусловия	Пользователь авторизован (валидный Bearer токен). Поездка с {tripId} существует, и пользователь имеет права на редактирование ('edit' или 'owner').
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId (ID поездки) Тело запроса (JSON): { "title": "...", "description": "..." } (новые валидные данные, соответствующие TripUpdate)
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать PUT-запрос с новыми данными в теле. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект Trip.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме Trip. - Обновленные поля в ответе содержат новые значения. - Поле updatedAt изменилось. - При последующем GET-запросе к этой поездке

	возвращаются обновленные данные.
Полученный результат	Статус код 200. Получен объект Trip с обновленными данными. Изменения подтверждены GET-запросом.

3.2.5 Удаление поездки (DELETE /trips/{tripId})

Параметр	**Описание**
Название теста	Успешное удаление поездки создателем
Цель	Проверить возможность удаления поездки ее создателем.
Эндпоинт	DELETE /trips/{tripId}
Предусловия	Пользователь авторизован (валидный Bearer токен). Поездка с {tripId} существует, и авторизованный пользователь является ее создателем.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId (ID поездки пользователя)
Шаги выполнения	1. Авторизоваться как создатель поездки. 2. Сформировать DELETE-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - При последующем GET-запросе к этому {tripId} возвращается ошибка 404 (Not Found).
Полученный результат	Статус код 204. Тело ответа пустое. Поездка недоступна через GET-запрос (возвращается 404).

3.2.6 Получение всех дней поездки (GET /trips/{tripId}/days)

Параметр	**Описание**
Название теста	Получение списка дней существующей поездки
Цель	Проверить успешное получение списка всех дней, запланированных для конкретной поездки, к которой

	пользователь имеет доступ.
Эндпоинт	GET /trips/{tripId}/days
Предусловия	Пользователь авторизован (валидный Bearer токен). Поездка с {tripId} существует и содержит запланированные дни. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId (валидный UUID существующей поездки).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips/{tripId}/days. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив объектов TripDay.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа является массивом. - Структура элементов в массиве соответствует схеме TripDay. - Поле tripId у каждого дня в ответе совпадает с запрошенным.
Полученный результат	Статус код 200. Получен массив объектов TripDay для указанной поездки.

3.2.7 Добавление нового дня к поездке (POST /trips/{tripId}/days)

Параметр	**Описание**
Название теста	Успешное добавление нового дня к поездке
Цель	Проверить возможность добавления нового дня к существующей поездке пользователем с правом редактирования.
Эндпоинт	POST /trips/{tripId}/days
Предусловия	Пользователь авторизован (валидный Bearer токен). Поездка с {tripId} существует. Пользователь имеет

	права на редактирование поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId (ID поездки) Тело запроса (JSON): { "date": "..." } (валидные данные, соответствующие TripDayCreate)
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать POST-запрос с валидным телом. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит созданный объект TripDay.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект, соответствующий схеме TripDay. - Поле tripId в ответе совпадает с запрошенным. - Поле date в ответе совпадает с отправленным. - День действительно появился в списке (GET /trips/{tripId}/days).
Полученный результат	Статус код 201. Получен корректный объект TripDay. День добавлен и доступен через GET-запрос.

3.2.8 Получение конкретного дня поездки (GET /trips/{tripId}/days/{dayId})

Параметр	**Описание**
Название теста	Получение деталей конкретного дня поездки
Цель	Проверить успешное получение деталей конкретного дня в поездке по его ID.
Эндпоинт	GET /trips/{tripId}/days/{dayId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует внутри этой поездки. Пользователь имеет доступ к поездке.
Тип	Позитивный

тестирования	
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId (ID поездки), dayId (ID дня).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос с указанием tripId и dayId. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект TripDay.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме TripDay. - Поля id и tripId в ответе совпадают с запрошенными.
Полученный результат	Статус код 200. Получен корректный объект TripDay с запрошенными ID.

3.2.9 Обновление конкретного дня поездки (PUT /trips/{tripId}/days/{dayId})

Параметр	**Описание**
Название теста	Успешное обновление дня поездки
Цель	Проверить возможность обновления данных конкретного дня поездки пользователем с правом редактирования.
Эндпоинт	PUT /trips/{tripId}/days/{dayId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует. Пользователь имеет права на редактирование поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, dayId Тело запроса (JSON): { "note": "...", "dayNumber": ... } (валидные данные TripDayUpdate)

Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать PUT-запрос с новыми данными дня. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект TripDay.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект TripDay с обновленными полями. - Поле updatedAt изменилось. - При последующем GET-запросе к этому дню возвращаются обновленные данные.
Полученный результат	Статус код 200. Получен объект TripDay с обновленными данными. Изменения подтверждены GET-запросом.

3.2.10 Удаление конкретного дня из поездки (DELETE /trips/{tripId}/days/{dayId})

Параметр	**Описание**
Название теста	Успешное удаление дня из поездки
Цель	Проверить возможность удаления конкретного дня из поездки пользователем с правом редактирования.
Эндпоинт	DELETE /trips/{tripId}/days/{dayId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует. Пользователь имеет права на редактирование поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, dayId.
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать DELETE-запрос.

	3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - При последующем GET-запросе к этому {dayId} возвращается 404. - День отсутствует в списке (GET /trips/{tripId}/days).
Полученный результат	Статус код 204. Тело ответа пустое. День удален и недоступен через GET-запросы.

3.2.11 Получение всех мест для конкретного дня (GET /trips/{tripId}/days/{dayId}/places)

Параметр	**Описание**
Название теста	Получение списка мест для дня поездки
Цель	Проверить успешное получение списка всех мест, запланированных на конкретный день поездки.
Эндпоинт	GET /trips/{tripId}/days/{dayId}/places
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует и содержит запланированные места. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, dayId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips/{tripId}/days/{dayId}/places. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив объектов Place.
Проверка	Проверить, что:

	- Статус код ответа 200. - Тело ответа является массивом. - Структура элементов в массиве соответствует схеме Place.
Полученный результат	Статус код 200. Получен массив объектов Place для указанного дня поездки.

3.2.12 Добавление нового места ко дню (POST /trips/{tripId}/days/{dayId}/places)

Параметр	**Описание**
Название теста	Успешное добавление нового места ко дню поездки
Цель	Проверить возможность добавления нового места к конкретному дню поездки пользователем с правом редактирования.
Эндпоинт	POST /trips/{tripId}/days/{dayId}/places
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует. Пользователь имеет права на редактирование поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, dayId Тело запроса (JSON): { "name": "..." } (валидные данные PlaceCreate)
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать POST-запрос с валидным телом. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит созданный объект Place.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект, соответствующий схеме Place. - Поле name в ответе совпадает с отправленным. - Место действительно появилось в списке (GET

	/trips/{tripId}/days/{dayId}/places).
Полученный результат	Статус код 201. Получен корректный объект Place. Место добавлено и доступно через GET-запрос.

3.2.13 Получение деталей конкретного места (GET /trips/{tripId}/days/{dayId}/places/{placeId})

Параметр	**Описание**
Название теста	Получение деталей конкретного места в дне поездки
Цель	Проверить успешное получение деталей конкретного места в дне поездки по его ID.
Эндпоинт	GET /trips/{tripId}/days/{dayId}/places/{placeId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует. Место с {placeId} существует в этом дне. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, dayId, placeId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос с указанием tripId, dayId, placeId. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект Place.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме Place. - Поле id в ответе совпадает с запрошенным placeId.
Полученный результат	Статус код 200. Получен корректный объект Place с запрошенным ID.

3.2.14 Обновление конкретного места (PUT /trips/{tripId}/days/{dayId}/places/{placeId})

Параметр	Описание
Название теста	Успешное обновление места в дне поездки
Цель	Проверить возможность обновления данных конкретного места в дне поездки пользователем с правом редактирования.
Эндпоинт	PUT /trips/{tripId}/days/{dayId}/places/{placeId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует. Место с {placeId} существует. Пользователь имеет права на редактирование поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, dayId, placeId Тело запроса (JSON): { "name": "...", "address": "..." } (валидные данные PlaceUpdate)
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать PUT-запрос с новыми данными места. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект Place.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект Place с обновленными полями. - Поле updatedAt изменилось. - При последующем GET-запросе к этому месту возвращаются обновленные данные.
Полученный результат	Статус код 200. Получен объект Place с обновленными данными. Изменения подтверждены GET-запросом.

3.2.15 Удаление места из дня (DELETE /trips/{tripId}/days/{dayId}/places/{placeId})

Параметр	Описание
Название теста	Успешное удаление места из дня поездки
Цель	Проверить возможность удаления конкретного места из дня поездки пользователем с правом редактирования.
Эндпоинт	DELETE /trips/{tripId}/days/{dayId}/places/{placeId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. День с {dayId} существует. Место с {placeId} существует. Пользователь имеет права на редактирование поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, dayId, placeId.
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать DELETE-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - При последующем GET-запросе к этому {placeId} возвращается 404. - Место отсутствует в списке (GET /trips/{tripId}/days/{dayId}/places).
Полученный результат	Статус код 204. Тело ответа пустое. Место удалено и недоступно через GET-запросы.

3.2.16 Получение всех файлов, прикрепленных к поездке (GET /trips/{tripId}/files)

Параметр	Описание
Название теста	Получение списка файлов поездки

Цель	Проверить успешное получение списка всех файлов, прикрепленных к конкретной поездке.
Эндпоинт	GET /trips/{tripId}/files
Предусловия	Пользователь авторизован. Поездка с {tripId} существует и содержит прикрепленные файлы. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips/{tripId}/files. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив объектов File.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа является массивом. - Структура элементов в массиве соответствует схеме File. - Поле tripId у каждого файла совпадает с запрошенным.
Полученный результат	Статус код 200. Получен массив объектов File для указанной поездки.

3.2.17 Прикрепление файла к поездке (POST /trips/{tripId}/files)

Параметр	**Описание**
Название теста	Успешное прикрепление файла к поездке
Цель	Проверить возможность загрузки и прикрепления файла к поездке пользователем с правом редактирования.
Эндпоинт	POST /trips/{tripId}/files
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. Пользователь имеет права на

	редактирование поездки. Файл для загрузки готов.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId Тело запроса (multipart/form-data): поле file (бинарные данные файла), поле description (опционально, строка)
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать POST-запрос с файлом и описанием. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит объект File, описывающий загруженный файл.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект File с полями id, filename, url, size, mimeType, tripId. - Файл действительно появился в списке (GET /trips/{tripId}/files).
Полученный результат	Статус код 201. Получен корректный объект File. Файл прикреплен и доступен через GET-запрос.

3.2.18 Получение (скачивание) конкретного файла (GET /trips/{tripId}/files/{fileId})

Параметр	**Описание**
Название теста	Успешное скачивание файла поездки
Цель	Проверить возможность скачивания конкретного файла, прикрепленного к поездке.
Эндпоинт	GET /trips/{tripId}/files/{fileId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. Файл с {fileId} существует и прикреплен к поездке. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный

Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, fileId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос с указанием tripId и fileId. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит бинарные данные файла (application/octet-stream).
Проверка	Проверить, что: - Статус код ответа 200. - Content-Type ответа application/octet-stream (или соответствует mimeType файла). - Тело ответа содержит непустые бинарные данные. - Размер скачанного файла соответствует полю size в метаданных файла.
Полученный результат	Статус код 200. Получены бинарные данные файла с корректным Content-Type.

3.2.19 Удаление файла из поездки (DELETE /trips/{tripId}/files/{fileId})

Параметр	**Описание**
Название теста	Успешное удаление файла из поездки
Цель	Проверить возможность удаления файла из поездки пользователем с правом редактирования.
Эндпоинт	DELETE /trips/{tripId}/files/{fileId}
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. Файл с {fileId} существует. Пользователь имеет права на редактирование поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, fileId.
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать DELETE-запрос.

	3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - При последующем GET-запросе к /trips/{tripId}/files/{fileId} возвращается 404. - Файл отсутствует в списке (GET /trips/{tripId}/files).
Полученный результат	Статус код 204. Тело ответа пустое. Файл удален и недоступен через GET-запросы.

3.2.20 Получение всех файлов, прикрепленных к месту (GET /trips/{tripId}/places/{placeId}/files)

Параметр	**Описание**
Название теста	Получение списка файлов места
Цель	Проверить успешное получение списка всех файлов, прикрепленных к конкретному месту в рамках поездки.
Эндпоинт	GET /trips/{tripId}/places/{placeId}/files
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. Место с {placeId} существует в рамках поездки и содержит прикрепленные файлы. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, placeId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips/{tripId}/places/{placeId}/files. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив объектов File.

Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа является массивом. - Структура элементов в массиве соответствует схеме File. - Поля tripId и placeId у каждого файла совпадают с запрошенными.
Полученный результат	Статус код 200. Получен массив объектов File для указанного места.

3.2.21 Прикрепление файла к месту (POST /trips/{tripId}/places/{placeId}/files)

Параметр	**Описание**
Название теста	Успешное прикрепление файла к месту
Цель	Проверить возможность загрузки и прикрепления файла к конкретному месту в поездке пользователем с правом редактирования.
Эндпоинт	POST /trips/{tripId}/places/{placeId}/files
Предусловия	Пользователь авторизован. Поездка с {tripId} существует. Место с {placeId} существует. Пользователь имеет права на редактирование поездки. Файл для загрузки готов.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, placeId Тело запроса (multipart/form-data): поле file, поле description
Шаги выполнения	1. Авторизоваться как пользователь с правом редактирования. 2. Сформировать POST-запрос с файлом и описанием. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит объект File, описывающий загруженный файл.

Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект File с корректными tripId и placeId. - Файл действительно появился в списке (GET /trips/{tripId}/places/{placeId}/files).
Полученный результат	Статус код 201. Получен корректный объект File. Файл прикреплен к месту и доступен через GET-запрос.

3.2.22 Получение данных карты для поездки (GET /trips/{tripId}/map)

Параметр	**Описание**
Название теста	Получение данных для карты поездки
Цель	Проверить успешное получение географических данных всех мест в поездке для отображения на карте.
Эндпоинт	GET /trips/{tripId}/map
Предусловия	Пользователь авторизован. Поездка с {tripId} существует и содержит дни с местами, имеющими координаты. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips/{tripId}/map. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект TripMapData.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа соответствует схеме TripMapData. - Содержит tripId, массив days. - Каждый день содержит

	массив places с id, name, coordinates, order.
Полученный результат	Статус код 200. Получен корректный объект TripMapData со структурой для отображения карты.

3.2.23 Поделиться поездкой с другим пользователем (POST /trips/{tripId}/share)

Параметр	**Описание**
Название теста	Успешное предоставление доступа к поездке другому пользователю
Цель	Проверить возможность предоставления доступа (share) к поездке другому пользователю с указанием прав (view/edit).
Эндпоинт	POST /trips/{tripId}/share
Предусловия	Пользователь авторизован (создатель поездки). Поездка с {tripId} существует. Пользователь, которому предоставляется доступ (recipient), существует в системе.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId Тело запроса (JSON): { "recipient": "user@example.com", "permission": "view" }
Шаги выполнения	1. Авторизоваться как создатель поездки. 2. Сформировать POST-запрос с email/username получателя и правами доступа. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит сообщение об успехе и shareId.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит message и shareId (UUID). - В списке коллабораторов (GET /trips/{tripId}/collaborators) появился новый пользователь с указанными правами.

Полученный результат	Статус код 200. Получено сообщение об успехе и shareId. Доступ предоставлен, пользователь виден в коллабораторах.

3.2.24 Удаление доступа для пользователя (DELETE /trips/{tripId}/share/{shareId})

Параметр	**Описание**
Название теста	Успешное удаление доступа к поездке для пользователя
Цель	Проверить возможность отзыва доступа к поездке у конкретного пользователя создателем поездки.
Эндпоинт	DELETE /trips/{tripId}/share/{shareId}
Предусловия	Пользователь авторизован (создатель поездки). Поездка с {tripId} существует. Запись о доступе с {shareId} существует для этой поездки.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: tripId, shareId.
Шаги выполнения	1. Авторизоваться как создатель поездки. 2. Сформировать DELETE-запрос с указанием tripId и shareId. 3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - Пользователь, соответствующий {shareId}, отсутствует в списке коллабораторов (GET /trips/{tripId}/collaborators).
Полученный результат	Статус код 204. Тело ответа пустое. Доступ для пользователя отозван.

3.2.25 Получение всех коллабораторов поездки (GET /trips/{tripId}/collaborators)

Параметр	Описание
Название теста	Получение списка коллабораторов поездки
Цель	Проверить успешное получение списка всех пользователей, имеющих доступ к конкретной поездке.
Эндпоинт	GET /trips/{tripId}/collaborators
Предусловия	Пользователь авторизован. Поездка с {tripId} существует и к ней предоставлен доступ другим пользователям. Пользователь имеет доступ к поездке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: tripId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /trips/{tripId}/collaborators. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив объектов Collaborator.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа является массивом. - Структура элементов в массиве соответствует схеме Collaborator, включая user и permission.
Полученный результат	Статус код 200. Получен массив объектов Collaborator для указанной поездки.

3.2.26 Получение всех списков дел (GET /todo-lists)

Параметр	Описание
Название теста	Получение списка дел пользователя (с пагинацией)
Цель	Проверить успешное получение списка всех списков дел, созданных авторизованным пользователем, с пагинацией.

Эндпоинт	GET /todo-lists
Предусловия	Пользователь авторизован. У пользователя существуют списки дел.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Query parameters: limit=10, offset=0 (пример).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /todo-lists с параметрами пагинации. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с total и todoLists (массив TodoListSummary).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит total и todoLists. - Структура элементов в todoLists соответствует TodoListSummary. - Количество элементов не превышает limit.
Полученный результат	Статус код 200. Ответ содержит корректные поля total и todoLists. Структура и пагинация соответствуют ожидаемой.

3.2.27 Создание нового списка дел (POST /todo-lists)

Параметр	**Описание**
Название теста	Успешное создание нового списка дел
Цель	Проверить возможность создания нового списка дел авторизованным пользователем.
Эндпоинт	POST /todo-lists
Предусловия	Пользователь авторизован.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Тело запроса (JSON): { "title": "..." } (валидные данные TodoListCreate)

Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос с валидным телом. 3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит созданный объект TodoList.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект TodoList. - Поле title соответствует отправленному. - Поле userId соответствует ID авторизованного пользователя. - Список дел появился в общем списке (GET /todo-lists).
Полученный результат	Статус код 201. Получен корректный объект TodoList. Список дел доступен через GET-запрос.

3.2.28 Получение доступных шаблонов списков дел (GET /todo-lists/templates)

Параметр	**Описание**
Название теста	Получение списка шаблонов списков дел
Цель	Проверить успешное получение списка всех доступных шаблонов для создания списков дел.
Эндпоинт	GET /todo-lists/templates
Предусловия	Пользователь авторизован. Существуют предопределенные шаблоны в системе.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token>
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /todo-lists/templates. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив объектов TodoListTemplate.
Проверка	Проверить, что:

	- Статус код ответа 200. - Тело ответа является массивом. - Структура элементов в массиве соответствует схеме TodoListTemplate.
Полученный результат	Статус код 200. Получен массив объектов TodoListTemplate.

3.2.29 Получение деталей списка дел (GET /todo-lists/{todoListId})

Параметр	**Описание**
Название теста	Получение деталей существующего списка дел
Цель	Проверить успешное получение полной информации о конкретном списке дел по его ID.
Эндпоинт	GET /todo-lists/{todoListId}
Предусловия	Пользователь авторизован. Список дел с {todoListId} существует и принадлежит пользователю.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: todoListId (валидный UUID списка дел пользователя).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /todo-lists/{todoListId}. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект TodoList с полной информацией.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект TodoList. - Поле id совпадает с запрошенным todoListId. - Содержит массив items.
Полученный результат	Статус код 200. Получен корректный объект TodoList с запрошенным ID и деталями списка.

3.2.30 Обновление деталей списка дел (PUT /todo-lists/{todoListId})

Параметр	Описание
Название теста	Успешное обновление данных списка дел
Цель	Проверить возможность обновления информации списка дел его владельцем.
Эндпоинт	PUT /todo-lists/{todoListId}
Предусловия	Пользователь авторизован. Список дел с {todoListId} существует и принадлежит пользователю.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: todoListId Тело запроса (JSON): { "title": "...", "description": "..." } (валидные данные TodoListUpdate)
Шаги выполнения	1. Авторизоваться как владелец списка. 2. Сформировать PUT-запрос с новыми данными. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект TodoList.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект TodoList с обновленными полями. - Поле updatedAt изменилось. - При последующем GET-запросе возвращаются обновленные данные.
Полученный результат	Статус код 200. Получен объект TodoList с обновленными данными. Изменения подтверждены GET-запросом.

3.2.31 Удаление списка дел (DELETE /todo-lists/{todoListId})

Параметр	Описание
Название теста	Успешное удаление списка дел
Цель	Проверить возможность удаления списка дел его владельцем.

Эндпоинт	DELETE /todo-lists/{todoListId}
Предусловия	Пользователь авторизован. Список дел с {todoListId} существует и принадлежит пользователю.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: todoListId.
Шаги выполнения	1. Авторизоваться как владелец списка. 2. Сформировать DELETE-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - При последующем GET-запросе к {todoListId} возвращается 404.
Полученный результат	Статус код 204. Тело ответа пустое. Список дел удален и недоступен через GET-запрос.

3.2.32 Получение всех элементов списка дел (GET /todo-lists/{todoListId}/items)

Параметр	**Описание**
Название теста	Получение списка элементов существующего списка дел
Цель	Проверить успешное получение списка всех элементов для конкретного списка дел.
Эндпоинт	GET /todo-lists/{todoListId}/items
Предусловия	Пользователь авторизован. Список дел с {todoListId} существует, принадлежит пользователю и содержит элементы.
Тип тестирования	Позитивный

Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: todoListId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /todo-lists/{todoListId}/items. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит массив объектов TodoItem.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа является массивом. - Структура элементов соответствует схеме TodoItem. - Поле listId у каждого элемента совпадает с запрошенным.
Полученный результат	Статус код 200. Получен массив объектов TodoItem для указанного списка дел.

3.2.33 Добавление нового элемента в список дел (POST /todo-lists/{todoListId}/items)

Параметр	**Описание**
Название теста	Успешное добавление нового элемента в список дел
Цель	Проверить возможность добавления нового элемента в существующий список дел.
Эндпоинт	POST /todo-lists/{todoListId}/items
Предусловия	Пользователь авторизован. Список дел с {todoListId} существует и принадлежит пользователю.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: todoListId Тело запроса (JSON): { "content": "..." } (валидные данные TodoItemCreate)
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос с валидным телом.

	3. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит созданный объект TodoItem.
Проверка	Проверить, что: - Статус код ответа 201. - Тело ответа содержит объект TodoItem. - Поле listId совпадает с todoListId. - Поле content совпадает с отправленным. - Элемент появился в списке (GET /todo-lists/{todoListId}/items).
Полученный результат	Статус код 201. Получен корректный объект TodoItem. Элемент добавлен и доступен через GET-запрос.

3.2.34 Обновление конкретного элемента списка дел (PUT /todo-lists/{todoListId}/items/{itemId})

Параметр	**Описание**
Название теста	Успешное обновление элемента списка дел
Цель	Проверить возможность обновления данных конкретного элемента списка дел.
Эндпоинт	PUT /todo-lists/{todoListId}/items/{itemId}
Предусловия	Пользователь авторизован. Список дел {todoListId} существует. Элемент {itemId} существует в этом списке. Пользователь - владелец списка.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: todoListId, itemId Тело запроса (JSON): { "content": "...", "completed": true } (валидные данные TodoItemUpdate)
Шаги выполнения	1. Авторизоваться. 2. Сформировать PUT-запрос с новыми данными элемента. 3. Отправить запрос.
Ожидаемый	Code 200. Ответ содержит обновленный

результат	объект TodoItem.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект TodoItem с обновленными полями. - Поле updatedAt изменилось. - При последующем GET-запросе к элементам списка возвращаются обновленные данные.
Полученный результат	Статус код 200. Получен объект TodoItem с обновленными данными. Изменения подтверждены GET-запросом.

3.2.35 Удаление конкретного элемента списка дел (DELETE /todo-lists/{todoListId}/items/{itemId})

Параметр	**Описание**
Название теста	Успешное удаление элемента из списка дел
Цель	Проверить возможность удаления конкретного элемента из списка дел.
Эндпоинт	DELETE /todo-lists/{todoListId}/items/{itemId}
Предусловия	Пользователь авторизован. Список дел {todoListId} существует. Элемент {itemId} существует. Пользователь - владелец списка.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: todoListId, itemId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать DELETE-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 204. Тело ответа отсутствует.
Проверка	Проверить, что: - Статус код ответа 204. - Тело ответа пустое. - Элемент отсутствует в списке (GET /todo-

	lists/{todoListId}/items).
Полученный результат	Статус код 204. Тело ответа пустое. Элемент удален из списка.

3.2.36 Переключение статуса выполнения элемента списка дел (PUT /todo-lists/{todoListId}/items/{itemId}/toggle)

Параметр	**Описание**
Название теста	Успешное переключение статуса выполнения элемента
Цель	Проверить возможность изменения статуса completed для элемента списка дел.
Эндпоинт	PUT /todo-lists/{todoListId}/items/{itemId}/toggle
Предусловия	Пользователь авторизован. Список дел {todoListId} существует. Элемент {itemId} существует (например, completed: false). Пользователь - владелец списка.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameters: todoListId, itemId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать PUT-запрос к /toggle. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект TodoItem с измененным статусом completed.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект TodoItem. - Значение поля completed инвертировано по сравнению с исходным. - Поле updatedAt изменилось.
Полученный результат	Статус код 200. Получен объект TodoItem с переключенным статусом completed.

3.2.37 Генерация списка дел с помощью AI (POST /todo-lists/ai-generate)

Параметр	Описание
Название теста	Успешная генерация списка дел с помощью AI
Цель	Проверить возможность генерации нового списка дел на основе входных параметров с использованием AI.
Эндпоинт	POST /todo-lists/ai-generate
Предусловия	Пользователь авторизован. Сервис AI доступен и настроен.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Тело запроса (JSON): { "destination": "Paris", "purpose": "tourism", "duration": 7 } (валидные параметры)
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос с параметрами для AI. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект TodoList, сгенерированный AI, включая элементы (items).
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект TodoList. - Список содержит релевантные элементы (items), основанные на входных параметрах. - Список дел сохранен и доступен (GET /todo-lists/{newListId}).
Полученный результат	Статус код 200. Получен релевантный объект TodoList, сгенерированный AI. Список сохранен.

3.2.38 Получение всех уведомлений (GET /notifications)

Параметр	Описание
Название теста	Получение списка уведомлений пользователя (с фильтрацией)

Цель	Проверить успешное получение списка уведомлений авторизованного пользователя с фильтрацией по статусу прочтения и пагинацией.
Эндпоинт	GET /notifications
Предусловия	Пользователь авторизован. У пользователя существуют уведомления (прочитанные и непрочитанные).
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Query parameters: read=false, limit=15, offset=0 (пример).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /notifications с параметрами. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с total и notifications (массив Notification).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит total и notifications. - Структура элементов соответствует Notification. - Если read указан, все уведомления в ответе имеют соответствующий статус. - Количество не превышает limit.
Полученный результат	Статус код 200. Ответ содержит корректные поля total и notifications. Структура, фильтрация и пагинация соответствуют ожидаемой.

3.2.39 Отметка уведомления как прочитанного (PUT /notifications/{notificationId}/read)

Параметр	**Описание**
Название теста	Успешная отметка уведомления как прочитанного
Цель	Проверить возможность отметки конкретного непрочитанного уведомления как прочитанного.
Эндпоинт	PUT /notifications/{notificationId}/read

Предусловия	Пользователь авторизован. Уведомление с {notificationId} существует, принадлежит пользователю и имеет статус read: false.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: notificationId.
Шаги выполнения	1. Авторизоваться. 2. Сформировать PUT-запрос. 3. Отправить запрос.
Ожидаемый результат	Code 200. Тело ответа отсутствует или содержит сообщение об успехе.
Проверка	Проверить, что: - Статус код ответа 200. - При последующем GET-запросе к /notifications, данное уведомление имеет статус read: true.
Полученный результат	Статус код 200. Уведомление успешно помечено как прочитанное.

3.2.40 Отметка всех уведомлений как прочитанных (PUT /notifications/read-all)

Параметр	**Описание**
Название теста	Успешная отметка всех уведомлений как прочитанных
Цель	Проверить возможность отметки всех уведомлений пользователя как прочитанных одним запросом.
Эндпоинт	PUT /notifications/read-all
Предусловия	Пользователь авторизован. У пользователя есть непрочитанные уведомления.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token>
Шаги	1. Авторизоваться.

выполнения	2. Сформировать PUT-запрос к /notifications/read-all. 3. Отправить запрос.
Ожидаемый результат	Code 200. Тело ответа отсутствует или содержит сообщение об успехе.
Проверка	Проверить, что: - Статус код ответа 200. - При последующем GET-запросе к /notifications с фильтром read=false, возвращается пустой список или список без ранее непрочитанных уведомлений.
Полученный результат	Статус код 200. Все уведомления пользователя успешно помечены как прочитанные.

3.2.41 Получение настроек уведомлений (GET /notifications/settings)

Параметр	**Описание**
Название теста	Получение текущих настроек уведомлений
Цель	Проверить успешное получение текущих настроек уведомлений для авторизованного пользователя.
Эндпоинт	GET /notifications/settings
Предусловия	Пользователь авторизован. Существуют настройки уведомлений для пользователя.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token>
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /notifications/settings. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект NotificationSettings.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме NotificationSettings. - Поле userId совпадает с ID авторизованного пользователя.

Полученный результат	Статус код 200. Получен корректный объект NotificationSettings.

3.2.42 Обновление настроек уведомлений (PUT /notifications/settings)

Параметр	**Описание**
Название теста	Успешное обновление настроек уведомлений
Цель	Проверить возможность обновления настроек уведомлений для авторизованного пользователя.
Эндпоинт	PUT /notifications/settings
Предусловия	Пользователь авторизован.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Тело запроса (JSON): { "tripReminders": false, "reminderTime": 30 } (валидные данные NotificationSettingsUpdate)
Шаги выполнения	1. Авторизоваться. 2. Сформировать PUT-запрос с новыми настройками. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект NotificationSettings.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект NotificationSettings с обновленными значениями. - При последующем GET-запросе к /notifications/settings возвращаются обновленные настройки.
Полученный результат	Статус код 200. Получен объект NotificationSettings с обновленными данными. Изменения подтверждены GET-запросом.

3.3 Сервис auth

3.3.1 Вход в систему (POST /login)

Параметр	Описание
Название теста	Успешная авторизация пользователя
Цель	Проверить возможность успешной авторизации зарегистрированного и подтвержденного пользователя по email и паролю и получение токенов.
Эндпоинт	POST /login
Предусловия	Пользователь с указанным email зарегистрирован, его email подтвержден. Пароль корректен. API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "email": "user@example.com", "password": "correct_password", "deviceId": "optional_device_id" }
Шаги выполнения	1. Сформировать POST-запрос к /login с корректными email и паролем. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит accessToken, refreshToken, expiresIn и объект user (соответствующий UserInfo).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит непустые строки accessToken и refreshToken. - Поле expiresIn является числом. - Объект user содержит корректную информацию о пользователе (ID, email, username и т.д.).
Полученный результат	Статус код 200. Получены accessToken, refreshToken, expiresIn и корректная информация о пользователе.

3.3.2 Обновление токена (POST /refresh)

Параметр	Описание
Название теста	Успешное обновление токена доступа

Цель	Проверить возможность получения новой пары accessToken и refreshToken с использованием валидного refreshToken.
Эндпоинт	POST /refresh
Предусловия	Пользователь ранее успешно авторизовался и получил валидный, неистекший refreshToken. API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "refreshToken": "valid_refresh_token_string" }
Шаги выполнения	1. Сформировать POST-запрос к /refresh с валидным refreshToken. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит новые accessToken, refreshToken и expiresIn.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит новые, непустые строки accessToken и refreshToken. - Новые токены отличаются от предыдущих. - Поле expiresIn является числом.
Полученный результат	Статус код 200. Получены новые accessToken и refreshToken.

3.3.3 Выход из системы (POST /logout)

Параметр	**Описание**
Название теста	Успешный выход пользователя из системы
Цель	Проверить возможность инвалидации текущих токенов пользователя (на стороне сервера, если применимо).
Эндпоинт	POST /logout
Предусловия	Пользователь авторизован (имеет валидный accessToken).
Тип	Позитивный

тестирования	
Данные запроса	Заголовок: Authorization: Bearer <valid_access_token>
Шаги выполнения	1. Авторизоваться (получить токен). 2. Сформировать POST-запрос к /logout с Authorization заголовком. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит сообщение об успешном выходе.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит сообщение "Вы успешно вышли из системы" (или аналогичное). - Последующие запросы с использованным accessToken (и refreshToken, если он инвалидируется) возвращают ошибку 401.
Полученный результат	Статус код 200. Получено сообщение об успешном выходе. Токены инвалидированы (проверено последующим запросом).

3.3.4 Регистрация нового пользователя (POST /register)

Параметр	**Описание**
Название теста	Успешная регистрация нового пользователя
Цель	Проверить возможность создания новой учетной записи пользователя с уникальным email.
Эндпоинт	POST /register
Предусловия	Email, используемый для регистрации, еще не зарегистрирован в системе. API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "email": "new_user@example.com", "password": "strong_password", "username": "newbie", "firstName": "...", "lastName": "...", "deviceId": "..." }
Шаги	1. Сформировать POST-запрос к /register с валидными

выполнения	и уникальными данными. 2. Отправить запрос.
Ожидаемый результат	Code 201. Ответ содержит сообщение об успехе и userId созданного пользователя. Письмо для подтверждения email отправлено.
Проверка	Проверить, что: - Статус код ответа 201. - Ответ содержит сообщение о необходимости подтверждения email и userId (UUID). - На указанный email пришло письмо со ссылкой/кодом подтверждения. - Попытка входа (/login) до подтверждения email не удается (или удается с ограниченным доступом, в зависимости от логики).
Полученный результат	Статус код 201. Получено сообщение об успехе и userId. Письмо подтверждения отправлено.

3.3.5 Подтверждение email (POST /verify-email)

Параметр	**Описание**
Название теста	Успешное подтверждение email пользователя
Цель	Проверить возможность подтверждения email с использованием валидного токена из письма.
Эндпоинт	POST /verify-email
Предусловия	Пользователь зарегистрирован, но email еще не подтвержден. Получен валидный, неистекший токен подтверждения из email. API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "token": "valid_verification_token_string" }
Шаги выполнения	1. Получить токен из письма подтверждения. 2. Сформировать POST-запрос к /verify-email с этим токеном. 3. Отправить запрос.

Ожидаемый результат	Code 200. Ответ содержит сообщение об успехе, accessToken, refreshToken, expiresIn и user (с email Verified: true).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит сообщение об успехе и валидные токены (accessToken, refreshToken). - Объект user содержит поле emailVerified со значением true. - Пользователь теперь может успешно войти через /login.
Полученный результат	Статус код 200. Email успешно подтвержден, получены токены доступа. Пользователь может войти в систему.

3.3.6 Повторная отправка письма подтверждения (POST /resend-verification)

Параметр	**Описание**
Название теста	Успешная повторная отправка письма подтверждения
Цель	Проверить возможность запроса нового письма для подтверждения email для зарегистрированного, но не подтвержденного пользователя.
Эндпоинт	POST /resend-verification
Предусловия	Пользователь с указанным email зарегистрирован, но его email еще не подтвержден. API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "email": "unverified_user@example.com" }
Шаги выполнения	1. Сформировать POST-запрос к /resend-verification с email неподтвержденного пользователя. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит сообщение об успешной отправке письма.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит сообщение об отправке нового письма.

	- На указанный email пришло новое письмо с ссылкой/кодом подтверждения.
Полученный результат	Статус код 200. Получено сообщение об успехе. Новое письмо подтверждения отправлено.

3.3.7 Запрос на восстановление пароля (POST /forgot-password)

Параметр	**Описание**
Название теста	Успешный запрос на восстановление пароля
Цель	Проверить возможность инициировать процесс восстановления пароля для существующего пользователя по email.
Эндпоинт	POST /forgot-password
Предусловия	Пользователь с указанным email зарегистрирован в системе. API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "email": "registered_user@example.com" }
Шаги выполнения	1. Сформировать POST-запрос к /forgot-password с email зарегистрированного пользователя. 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит сообщение об отправке инструкций.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит сообщение об отправке инструкций на email. - На указанный email пришло письмо с кодом/ссылкой для восстановления пароля.
Полученный результат	Статус код 200. Получено сообщение об успехе. Письмо с инструкциями отправлено.

3.3.8 Проверка кода восстановления (POST /verify-reset-code)

Параметр	**Описание**
Название теста	Успешная проверка кода восстановления пароля

Цель	Проверить валидность кода восстановления, полученного из письма, и получение токена для сброса пароля.
Эндпоинт	POST /verify-reset-code
Предусловия	Пользователь запросил восстановление пароля. Получен валидный, неистекший код восстановления. API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "email": "registered_user@example.com", "code": "valid_reset_code" }
Шаги выполнения	1. Получить код из письма восстановления. 2. Сформировать POST-запрос к /verify-reset-code с email и кодом. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит сообщение о валидности кода и resetToken.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит сообщение о валидности кода и непустую строку resetToken.
Полученный результат	Статус код 200. Код подтвержден, получен resetToken.

3.3.9 Сброс пароля (POST /reset-password)

Параметр	**Описание**
Название теста	Успешный сброс (установка нового) пароля
Цель	Проверить возможность установки нового пароля с использованием валидного resetToken.
Эндпоинт	POST /reset-password
Предусловия	Пользователь успешно проверил код восстановления и получил валидный resetToken. Новый пароль соответствует требованиям безопасности. API доступно.

Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "resetToken": "valid_reset_token", "newPassword": "new_strong_password" }
Шаги выполнения	1. Получить resetToken после проверки кода. 2. Сформировать POST-запрос к /reset-password с resetToken и новым паролем. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит сообщение об успешном изменении пароля.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит сообщение об успехе. - Пользователь может войти в систему (/login) с новым паролем.
Полученный результат	Статус код 200. Пароль успешно изменен. Вход с новым паролем успешен.

3.3.10 Получение информации о текущем пользователе (GET /me)

Параметр	**Описание**
Название теста	Успешное получение данных текущего пользователя
Цель	Проверить возможность получения информации профиля для авторизованного пользователя.
Эндпоинт	GET /me
Предусловия	Пользователь авторизован (имеет валидный accessToken).
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <valid_access_token>
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /me с Authorization заголовком. 3. Отправить запрос.
Ожидаемый	Code 200. Ответ содержит объект UserInfo с

результат	данными текущего пользователя.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа соответствует схеме UserInfo. - Поля id, email, username и т.д. соответствуют данным авторизованного пользователя.
Полученный результат	Статус код 200. Получена корректная информация о профиле пользователя (UserInfo).

3.3.11 Обновление профиля пользователя (PUT /me)

Параметр	**Описание**
Название теста	Успешное обновление профиля пользователя
Цель	Проверить возможность изменения данных профиля (username, имя, фамилия, bio) для авторизованного пользователя.
Эндпоинт	PUT /me
Предусловия	Пользователь авторизован (имеет валидный accessToken).
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <valid_access_token> Тело запроса (JSON): { "username": "updated_user", "firstName": "NewName", "bio": "Updated bio" }
Шаги выполнения	1. Авторизоваться. 2. Сформировать PUT-запрос к /me с новыми данными профиля. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит обновленный объект UserInfo.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект UserInfo с обновленными значениями полей username, firstName, bio. - При последующем GET-запросе к /me возвращаются обновленные данные.

Полученный результат	Статус код 200. Профиль успешно обновлен, получен обновленный UserInfo. Изменения подтверждены GET-запросом.

3.3.12 Загрузка аватара (POST /me/avatar)

Параметр	**Описание**
Название теста	Успешная загрузка аватара пользователя
Цель	Проверить возможность загрузки файла изображения в качестве аватара для авторизованного пользователя.
Эндпоинт	POST /me/avatar
Предусловия	Пользователь авторизован. Файл изображения (например, jpg, png) допустимого размера готов к загрузке.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <valid_access_token> Тело запроса (multipart/form-data): поле avatar с бинарными данными файла изображения.
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос к /me/avatar с файлом изображения. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с avatarUrl.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит поле avatarUrl с валидным URL. - При последующем GET-запросе к /me поле avatarUrl содержит этот новый URL.
Полученный результат	Статус код 200. Аватар успешно загружен, получен avatarUrl. Новый URL аватара виден в профиле.

3.3.13 Изменение пароля (POST /me/change-password)

Параметр	**Описание**

Название теста	Успешное изменение пароля пользователя
Цель	Проверить возможность изменения пароля для авторизованного пользователя при указании верного текущего пароля.
Эндпоинт	POST /me/change-password
Предусловия	Пользователь авторизован. Пользователь знает свой текущий пароль. Новый пароль соответствует требованиям безопасности.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <valid_access_token> Тело запроса (JSON): { "currentPassword": "old_correct_password", "newPassword": "new_strong_password1" }
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос к /me/change-password с текущим и новым паролями. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит сообщение об успешном изменении пароля.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит сообщение об успехе. - Пользователь может выйти (/logout) и войти (/login) с новым паролем. - Вход со старым паролем не удается.
Полученный результат	Статус код 200. Пароль успешно изменен. Вход с новым паролем успешен, со старым - нет.

3.3.14 Получение анонимного токена (POST /anonymous-token)

Параметр	**Описание**
Название теста	Успешное получение анонимного токена
Цель	Проверить возможность получения временного токена для неавторизованных сессий с ограниченными правами.

Эндпоинт	POST /anonymous-token
Предусловия	API доступно.
Тип тестирования	Позитивный
Данные запроса	Тело запроса (JSON): { "deviceId": "optional_device_id_string" } (deviceId опционален)
Шаги выполнения	1. Сформировать POST-запрос к /anonymous-token (можно с deviceId или без). 2. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит anonymousToken и expiresIn.
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит непустую строку anonymousToken. - Поле expiresIn является числом. - Запросы к публичным ресурсам API с этим токеном в заголовке Authorization: Bearer <anonymousToken> выполняются успешно (если применимо).
Полученный результат	Статус код 200. Получен anonymousToken и expiresIn.

3.4 Сервис external

3.4.1 Поиск мест по названию (GET /places/search)

Параметр	**Описание**
Название теста	Успешный поиск мест по названию (с фильтрацией по категории и радиусу)
Цель	Проверить возможность поиска мест по текстовому запросу с использованием необязательных параметров: координаты, радиус, лимит и категория.
Эндпоинт	GET /places/search
Предусловия	Пользователь авторизован (валидный Bearer токен). Внешний картографический сервис доступен.
Тип тестирован	Позитивный

ия	
Данные запроса	Заголовок: Authorization: Bearer <token> Query parameters: query=Eiffel Tower, lat=48.85, lon=2.29, radius=1000, limit=10, category=attraction
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /places/search с параметрами поиска. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с total и places (массив объектов Place, соответствующих запросу).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит поля total (число) и places (массив). - Структура элементов в places соответствует схеме Place. - Количество элементов не превышает limit. - Найденные места релевантны запросу query, находятся в указанном радиусе (если lat/lon заданы) и соответствуют категории (если задана).
Полученный результат	Статус код 200. Ответ содержит список релевантных мест (places) и общее количество (total). Структура соответствует ожидаемой.

3.4.2 Получение детальной информации о месте (GET /places/{placeId})

Параметр	**Описание**
Название теста	Успешное получение детальной информации о месте
Цель	Проверить возможность получения подробной информации о конкретном месте по его ID из внешнего сервиса.
Эндпоинт	GET /places/{placeId}
Предусловия	Пользователь авторизован. ID места (placeId) валиден и существует во внешнем картографическом сервисе. Внешний сервис доступен.
Тип	Позитивный

тестирования	
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: placeId (валидный ID существующего места).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /places/{placeId}, подставив ID места. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект PlaceDetails с подробной информацией о месте.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа содержит объект, соответствующий схеме PlaceDetails. - Поле id в ответе совпадает с запрошенным placeId. - Присутствуют детальные поля (описание, телефон, часы работы, фото и т.д., если доступны).
Полученный результат	Статус код 200. Получен корректный объект PlaceDetails с запрошенным ID и подробной информацией о месте.

3.4.3 Автозаполнение для поиска мест (GET /places/autocomplete)

Параметр	**Описание**
Название теста	Успешное получение предложений для автозаполнения
Цель	Проверить возможность получения списка релевантных предложений (названий мест) при частичном вводе.
Эндпоинт	GET /places/autocomplete
Предусловия	Пользователь авторизован. Внешний картографический сервис доступен.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Query parameters: input=Louvre, lat=48.86, lon=2.33, limit=5

Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /places/autocomplete с частичным вводом и опциональными координатами. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с suggestions (массив объектов PlaceSuggestion).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит поле suggestions (массив). - Структура элементов в suggestions соответствует схеме PlaceSuggestion. - Предложения релевантны введенному input. - Количество предложений не превышает limit.
Полученный результат	Статус код 200. Ответ содержит список релевантных предложений (suggestions). Структура соответствует ожидаемой.

3.4.4 Поиск мест поблизости (GET /places/nearby)

Параметр	**Описание**
Название теста	Успешный поиск мест поблизости от заданных координат
Цель	Проверить возможность поиска мест в заданном радиусе от указанных координат с опциональной фильтрацией по категориям.
Эндпоинт	GET /places/nearby
Предусловия	Пользователь авторизован. Внешний картографический сервис доступен.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Query parameters: lat=40.7128, lon=-74.0060, radius=500, limit=15, categories=restaurant,cafe
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /places/nearby с координатами и другими параметрами. 3. Отправить запрос.

Ожидаемый результат	Code 200. Ответ содержит объект с total и places (массив объектов Place в указанном радиусе).
Проверка	Проверить, что: - Статус код ответа 200. - Ответ содержит поля total и places. - Структура элементов в places соответствует Place. - Количество элементов не превышает limit. - Все найденные места находятся в пределах указанного радиуса от заданных координат. - Если указаны categories, места соответствуют им.
Полученный результат	Статус код 200. Ответ содержит список мест (places) поблизости и общее количество (total). Структура и фильтрация соответствуют ожидаемой.

3.4.5 Генерация списка вещей для путешествия (POST /ai/packing-list)

Параметр	**Описание**
Название теста	Успешная генерация списка вещей с помощью ИИ
Цель	Проверить возможность генерации персонализированного списка вещей для упаковки на основе параметров поездки с использованием ИИ.
Эндпоинт	POST /ai/packing-list
Предусловия	Пользователь авторизован. Внешний сервис ИИ доступен и настроен.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Тело запроса (JSON): { "destination": "Thailand", "duration": 14, "season": "summer", "travelType": "beach", ... } (валидные данные PackingListRequest)
Шаги выполнения	1. Авторизоваться. 2. Сформировать POST-запрос к /ai/packing-list с параметрами поездки. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект PackingListResponse с категориями вещей,

	предметами и рекомендациями.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа соответствует схеме PackingListResponse. - Ответ содержит поле categories (массив категорий), где каждая категория содержит массив items. - Структура items соответствует описанию (name, description, priority, quantity). - Сгенерированный список релевантен входным параметрам.
Полученный результат	Статус код 200. Получен релевантный список вещей (PackingListResponse), сгенерированный ИИ. Структура соответствует ожидаемой.

3.4.6 Получение доступных шаблонов списков вещей (GET /ai/packing-list/templates)

Параметр	**Описание**
Название теста	Успешное получение списка шаблонов списков вещей
Цель	Проверить возможность получения списка предопределенных шаблонов для списков вещей.
Эндпоинт	GET /ai/packing-list/templates
Предусловия	Пользователь авторизован. Существуют предопределенные шаблоны в системе.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token>
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /ai/packing-list/templates. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект с templates (массив объектов PackingListTemplate).
Проверка	Проверить, что: - Статус код ответа 200.

	- Тело ответа содержит поле templates (массив). - Структура элементов в templates соответствует схеме PackingListTemplate.
Полученный результат	Статус код 200. Получен массив доступных шаблонов списков вещей (PackingListTemplate).

3.4.7 Получение содержимого шаблона списка вещей (GET /ai/packing-list/template/{templateId})

Параметр	**Описание**
Название теста	Успешное получение содержимого конкретного шаблона списка вещей
Цель	Проверить возможность получения полного содержимого (категорий и предметов) для конкретного шаблона по его ID.
Эндпоинт	GET /ai/packing-list/template/{templateId}
Предусловия	Пользователь авторизован. Шаблон с указанным templateId существует в системе.
Тип тестирования	Позитивный
Данные запроса	Заголовок: Authorization: Bearer <token> Path parameter: templateId (валидный ID существующего шаблона).
Шаги выполнения	1. Авторизоваться. 2. Сформировать GET-запрос к /ai/packing-list/template/{templateId}, подставив ID шаблона. 3. Отправить запрос.
Ожидаемый результат	Code 200. Ответ содержит объект PackingListTemplateContent с деталями шаблона.
Проверка	Проверить, что: - Статус код ответа 200. - Тело ответа соответствует схеме PackingListTemplateContent. - Поле id в ответе совпадает с запрошенным templateId. - Ответ содержит поле categories с массивом

	категорий и предметов внутри них.
Полученный результат	Статус код 200. Получен корректный объект PackingListTemplateContent с содержимым запрошенного шаблона.

4 Выявленные дефекты

Дефектов не обнаружено в ходе тестирования основных сценариев.

5 Общая оценка

- API работает стабильно
- Все базовые функции доступны и работают корректно
- Ошибки обрабатываются правильно