

Отчет: Система совместного планирования путешествий TravelTech

Проблема:

Современные путешественники, особенно семьи и группы, сталкиваются с трудностями при совместном планировании поездки:

- **Несогласованность:** Изменения в маршруте (рейсы, отели, активности) от одного участника не видны другим в реальном времени, что ведет к накладкам и конфликтам.
- **Оффлайн-работа:** Участники часто вносят правки в пути (в самолете, за границей без роуминга), и эти изменения теряются или создают путаницу при выходе в сеть.
- **Конфликты редактирования:** Несколько человек могут одновременно изменить или удалить один элемент плана, что приводит к потере данных или некорректному состоянию маршрута.

Запрос:

Спроектировать систему, которая обеспечивает:

- Согласованное состояние плана поездки для всей группы.
- Возможность для любого участника добавлять, редактировать или удалять элементы плана (рейсы, отели, активности).
- Доступность изменений для всех участников, включая текущее состояние и все предложенные варианты.
- Работу в условиях разных тайм-зон и оффлайн-сессий.
- Автоматическую предложить merge-стратегию при возникновении конфликтов одновременного редактирования.

API-спецификация (Swagger):

Данный API позволяет управлять совместным планом поездки в системе TravelTech. Он предоставляет методы для:

- ✓ Создания поездки и управления участниками.
- ✓ Добавления, редактирования и удаления элементов плана (с предложением и подтверждением).
- ✓ Синхронизации изменений между участниками, в том числе для оффлайн-клиентов.
- ✓ Получения актуального состояния плана, истории изменений и разрешения конфликтов.

Методы API:

Группа 1: Управление поездками и участниками

Метод	Эндпоинт	Описание
POST	/trips	Создать новую поездку
GET	/trips	Получить список своих поездок
GET	/trips/{tripId}	Получить полное текущее состояние поездки
DELETE	/trips/{tripId}	Удалить поездку (только создатель)
POST	/trips/{tripId}/members	Добавить/пригласить участника в поездку
DELETE	/trips/{tripId}/members/{userId}	Удалить участника из поездки
GET	/trips/{tripId}/members	Получить список участников поездки

Группа 2: Управление элементами плана

Метод	Эндпоинт	Описание
POST	/trips/{tripId}/items	Добавить новый элемент в план (со статусом PROPOSED)
PUT	/trips/{tripId}/items/{itemId}	Обновить существующий элемент плана
DELETE	/trips/{tripId}/items/{itemId}	Удалить элемент из плана
POST	/trips/{tripId}/items/{itemId}/confirm	Подтвердить предложенный элемент (PROPOSED → CONFIRMED)
POST	/trips/{tripId}/items/{itemId}/reject	Отклонить предложенный элемент (PROPOSED → REJECTED)
POST	/trips/{tripId}/items/{itemId}/book	Забронировать элемент через внешнего провайдера
GET	/trips/{tripId}/history	Получить историю изменений поездки

Группа 3: Поиск и каталог

Метод	Эндпоинт	Описание
GET	/search/flights	Поиск авиабилетов, поездов, автобусов (интеграция с внешними API)
GET	/search/hotels	Поиск отелей (интеграция с Booking/Agoda)
GET	/search/activities	Поиск развлечений и экскурсий
GET	/search/packages	Поиск готовых турпакетов

Группа 4: Рекомендации и персонализация

Метод	Эндпоинт	Описание
GET	/trips/{tripId}/recommendations	Получить персональные рекомендации для поездки
POST	/trips/{tripId}/items/{itemId}/like	Отметить "нравится" (для улучшения рекомендаций)
POST	/trips/{tripId}/items/{itemId}/dislike	Отметить "не нравится" (для улучшения рекомендаций)

Группа 5: Синхронизация и конфликты

Метод	Эндпоинт	Описание
POST	/trips/{tripId}/sync	Синхронизировать локальные изменения (отправка своих + получение чужих)
POST	/trips/{tripId}/conflicts/{conflictId}/resolve	Разрешить конфликт

Группа 6: Мониторинг статусов

Метод	Эндпоинт	Описание
GET	/external/status/flights/{flightId}	Проверить статус рейса (задержки, гейты)
GET	/external/status/bookings/{bookingId}	Проверить статус бронирования отеля

Группа 7: Платежи и документы

Метод	Эндпоинт	Описание
POST	/trips/{tripId}/items/{itemId}/pay	Инициировать оплату элемента
GET	/trips/{tripId}/payments	Получить историю платежей по поездке
POST	/trips/{tripId}/documents/passport	Загрузить шифрование паспортные данные
GET	/trips/{tripId}/documents	Получить документы по поездке (билеты, ваучеры)

OpenAPI-документ:

openapi: 3.0.0

info:

title: TravelTech Trip Planning API

description: API для совместного планирования путешествий с поддержкой оффлайн-работы и конфликт-менеджмента.

version: 1.0.0

contact:

name: Команда TravelTech

email: api@traveltech.com

license:

name: Проприетарная

url: https://traveltech.com/terms

servers:

- url: https://api.traveltech.com/v1

description: Production сервер

- url: https://sandbox.api.traveltech.com/v1

description: Тестовый сервер

tags:

- name: Trips

description: Управление поездками

- name: Items

description: Работа с элементами плана

- name: Search

description: Поиск услуг

- name: Recommendations

description: Персонализированные рекомендации

- name: Sync

description: Синхронизация и конфликты

- name: Monitoring

description: Мониторинг статусов

- name: Payments

description: Платежи и документы

paths:

===== Группа 1: Управление поездками и участниками =====

```
/trips:
  post:
    tags: [Trips]
    summary: Создать новую поездку
    operationId: createTrip
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/CreateTripRequest'
    responses:
      '201':
        description: Поездка успешно создана
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TripResponse'
      '400':
        description: Невалидные данные запроса

  get:
    tags: [Trips]
    summary: Получить список своих поездок
    operationId: getTrips
    parameters:
      - name: status
        in: query
        schema:
          type: string
          enum: [active, completed, archived]
        description: Фильтр по статусу поездки
    responses:
      '200':
        description: Список поездок пользователя
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/TripResponse'

/trips/{tripId}:
  get:
    tags: [Trips]
    summary: Получить полное текущее состояние поездки
    operationId: getTrip
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
```

```

        type: string
responses:
  '200':
    description: Полное состояние поездки
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/TripResponse'

delete:
  tags: [Trips]
  summary: Удалить поездку
  description: Только создатель поездки может ее удалить
  operationId: deleteTrip
  parameters:
    - name: tripId
      in: path
      required: true
      schema:
        type: string
  responses:
    '204':
      description: Поездка успешно удалена
    '403':
      description: Нет прав на удаление

/trips/{tripId}/members:
  post:
    tags: [Trips]
    summary: Добавить участника в поездку
    operationId: addMember
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AddMemberRequest'
    responses:
      '200':
        description: Участник успешно добавлен

  get:
    tags: [Trips]
    summary: Получить список участников поездки
    operationId: getMembers
    parameters:
      - name: tripId

```

```

    in: path
    required: true
    schema:
      type: string
  responses:
    '200':
      description: Список участников поездки
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/MemberResponse'

```

/trips/{tripId}/members/{userId}:

```

delete:
  tags: [Trips]
  summary: Удалить участника из поездки
  operationId: removeMember
  parameters:
    - name: tripId
      in: path
      required: true
      schema:
        type: string
    - name: userId
      in: path
      required: true
      schema:
        type: string
  responses:
    '204':
      description: Участник успешно удален
    '403':
      description: Нет прав на удаление участника

```

===== Группа 2: Управление элементами плана
=====

```

/trips/{tripId}/items:
  post:
    tags: [Items]
    summary: Добавить новый элемент в план
    description: Добавляет элемент со статусом PROPOSED
    (предложенный)
    operationId: addItem
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
    requestBody:

```

```

    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/CreateItemRequest'
  responses:
    '201':
      description: Элемент успешно добавлен
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ItemResponse'

/trips/{tripId}/items/{itemId}:
  put:
    tags: [Items]
    summary: Обновить существующий элемент плана
    operationId: updateItem
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
      - name: itemId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/UpdateItemRequest'
    responses:
      '200':
        description: Элемент успешно обновлен
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ItemResponse'

  delete:
    tags: [Items]
    summary: Удалить элемент из плана
    operationId: deleteItem
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string

```



```
- name: itemId
  in: path
  required: true
  schema:
    type: string
responses:
  '204':
    description: Элемент успешно удален

/trips/{tripId}/items/{itemId}/confirm:
  post:
    tags: [Items]
    summary: Подтвердить предложенный элемент
    description: Меняет статус элемента с PROPOSED на CONFIRMED
    operationId: confirmItem
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
      - name: itemId
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Элемент успешно подтвержден

/trips/{tripId}/items/{itemId}/reject:
  post:
    tags: [Items]
    summary: Отклонить предложенный элемент
    description: Меняет статус элемента с PROPOSED на REJECTED
    operationId: rejectItem
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
      - name: itemId
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Элемент успешно отклонен

/trips/{tripId}/items/{itemId}/book:
  post:
```

```

tags: [Items]
summary: Забронировать элемент через внешнего провайдера
description: Иницирует бронирование через API партнера
operationId: bookItem
parameters:
  - name: tripId
    in: path
    required: true
    schema:
      type: string
  - name: itemId
    in: path
    required: true
    schema:
      type: string
responses:
  '202':
    description: Бронирование принято в обработку
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/BookingResponse'

```

/trips/{tripId}/history:

```

get:
  tags: [Items]
  summary: Получить историю изменений поездки
  operationId: getHistory
  parameters:
    - name: tripId
      in: path
      required: true
      schema:
        type: string
    - name: limit
      in: query
      schema:
        type: integer
        default: 50
    description: Количество записей истории

```

```

responses:
  '200':
    description: История изменений
    content:
      application/json:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/HistoryEntry'

```

===== Группа 3: Поиск и каталог
=====

```
/search/flights:
  get:
    tags: [Search]
    summary: Поиск транспортных билетов
    description: Поиск авиабилетов, поездов, автобусов через
внешние API
    operationId: searchTransport
    parameters:
      - name: from
        in: query
        required: true
        schema:
          type: string
          description: Город отправления (код IATA или название)
      - name: to
        in: query
        required: true
        schema:
          type: string
          description: Город прибытия
      - name: date
        in: query
        required: true
        schema:
          type: string
          format: date
          description: Дата поездки (YYYY-MM-DD)
      - name: type
        in: query
        schema:
          type: string
          enum: [flight, train, bus, all]
          default: all
          description: Тип транспорта
      - name: passengers
        in: query
        schema:
          type: integer
          default: 1
          description: Количество пассажиров
    responses:
      '200':
        description: Результаты поиска
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/TransportOption'

/search/hotels:
  get:
    tags: [Search]
```

```
summary: Поиск отелей
operationId: searchHotels
parameters:
  - name: city
    in: query
    required: true
    schema:
      type: string
  - name: checkIn
    in: query
    required: true
    schema:
      type: string
      format: date
  - name: checkOut
    in: query
    required: true
    schema:
      type: string
      format: date
  - name: guests
    in: query
    schema:
      type: integer
      default: 2
responses:
  '200':
    description: Результаты поиска отелей
    content:
      application/json:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/HotelOption'
```

```
/search/activities:
  get:
    tags: [Search]
    summary: Поиск развлечений и экскурсий
    operationId: searchActivities
    parameters:
      - name: location
        in: query
        required: true
        schema:
          type: string
      - name: date
        in: query
        schema:
          type: string
          format: date
      - name: category
        in: query
```

```

    schema:
      type: string
      enum: [tour, show, sport, food, shopping]
  responses:
    '200':
      description: Результаты поиска активностей
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/ActivityOption'

```

```

/search/packages:
  get:
    tags: [Search]
    summary: Поиск готовых турпакетов
    operationId: searchPackages
    parameters:
      - name: destination
        in: query
        schema:
          type: string
      - name: duration
        in: query
        schema:
          type: integer
        description: Длительность в днях
      - name: priceFrom
        in: query
        schema:
          type: number
      - name: priceTo
        in: query
        schema:
          type: number

```

```

  responses:
    '200':
      description: Результаты поиска турпакетов
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/PackageOption'

```

===== Группа 4: Рекомендации и персонализация =====

```

/trips/{tripId}/recommendations:
  get:
    tags: [Recommendations]
    summary: Получить персональные рекомендации для поездки

```

```
operationId: getTripRecommendations
parameters:
  - name: tripId
    in: path
    required: true
    schema:
      type: string
  - name: category
    in: query
    schema:
      type: string
      enum: [hotels, activities, transport, restaurants]
  - name: limit
    in: query
    schema:
      type: integer
      default: 10
responses:
```

```
'200':
  description: Список рекомендаций
  content:
    application/json:
      schema:
        type: array
        items:
          $ref: '#/components/schemas/Recommendation'
```

/trips/{tripId}/items/{itemId}/like:

```
post:
  tags: [Recommendations]
  summary: Отметить "нравится"
  description: Учитывается для улучшения рекомендаций
  operationId: likeItem
  parameters:
    - name: tripId
      in: path
      required: true
      schema:
        type: string
    - name: itemId
      in: path
      required: true
      schema:
        type: string
  responses:
    '200':
      description: Отметка сохранена
```

/trips/{tripId}/items/{itemId}/dislike:

```
post:
  tags: [Recommendations]
  summary: Отметить "не нравится"
  description: Учитывается для улучшения рекомендаций
```

```

operationId: dislikeItem
parameters:
  - name: tripId
    in: path
    required: true
    schema:
      type: string
  - name: itemId
    in: path
    required: true
    schema:
      type: string
responses:
  '200':
    description: Отметка сохранена

```

```

# ===== Группа 5: Синхронизация и конфликты
=====

```

```

/trips/{tripId}/sync:
  post:
    tags: [Sync]
    summary: Синхронизировать локальные изменения
    description: Основной метод оффлайн-работы. Клиент отправляет
свои изменения и получает изменения других участников.
    operationId: syncTrip
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SyncRequest'
    responses:
      '200':
        description: Результат синхронизации
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SyncResponse'

/trips/{tripId}/conflicts/{conflictId}/resolve:
  post:
    tags: [Sync]
    summary: Разрешить конфликт
    operationId: resolveConflict
    parameters:
      - name: tripId

```

```

      in: path
      required: true
      schema:
        type: string
    - name: conflictId
      in: path
      required: true
      schema:
        type: string
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ResolveConflictRequest'
  responses:
    '200':
      description: Конфликт успешно разрешен

```

```

# ===== Группа 6: Мониторинг статусов
=====

```

```

/external/status/flights/{flightId}:
  get:
    tags: [Monitoring]
    summary: Проверить статус рейса
    description: Получить информацию о задержках, гейтах и статусе
рейса
    operationId: getFlightStatus
    parameters:
      - name: flightId
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Статус рейса
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FlightStatus'

```

```

/external/status/bookings/{bookingId}:
  get:
    tags: [Monitoring]
    summary: Проверить статус бронирования отеля
    operationId: getBookingStatus
    parameters:
      - name: bookingId
        in: path
        required: true
        schema:

```



```

        type: string
responses:
  '200':
    description: Статус бронирования
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/BookingStatus'

```

```

# ===== Группа 7: Платежи и документы
=====

```

```

/trips/{tripId}/items/{itemId}/pay:
  post:
    tags: [Payments]
    summary: Инициировать оплату элемента
    operationId: payItem
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
      - name: itemId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/PaymentRequest'
    responses:
      '200':
        description: Оплата инициирована
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/PaymentResponse'

```

```

/trips/{tripId}/payments:
  get:
    tags: [Payments]
    summary: Получить историю платежей по поездке
    operationId: getPayments
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string

```

```

responses:
  '200':
    description: История платежей
    content:
      application/json:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/PaymentHistoryEntry'

/trips/{tripId}/documents/passport:
  post:
    tags: [Payments]
    summary: Загрузить паспортные данные
    description: Данные шифруются при хранении
    operationId: uploadPassport
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/PassportData'
    responses:
      '201':
        description: Паспортные данные успешно загружены

/trips/{tripId}/documents:
  get:
    tags: [Payments]
    summary: Получить документы по поездке
    operationId: getDocuments
    parameters:
      - name: tripId
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Список документов
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/Document'

```

```
components:
  schemas:
    # Основные модели
    CreateTripRequest:
      type: object
      required:
        - name
        - startDate
      properties:
        name:
          type: string
          example: "Отпуск в Италии, июль 2024"
        startDate:
          type: string
          format: date
          example: "2024-07-15"
        endDate:
          type: string
          format: date
          example: "2024-07-25"
        description:
          type: string
          example: "Семейный отпуск в Риме и Флоренции"

    TripResponse:
      type: object
      properties:
        id:
          type: string
          example: "trip_123"
        name:
          type: string
        status:
          type: string
          enum: [planning, active, completed, archived]
          example: "planning"
        createdAt:
          type: string
          format: date-time
        updatedAt:
          type: string
          format: date-time
        items:
          type: array
          items:
            $ref: '#/components/schemas/ItemResponse'
        members:
          type: array
          items:
            $ref: '#/components/schemas/MemberResponse'

    AddMemberRequest:
      type: object
```

```
required:
  - userId
properties:
  userId:
    type: string
    example: "user_456"
  role:
    type: string
    enum: [editor, viewer]
    default: "editor"
```

```
MemberResponse:
  type: object
  properties:
    userId:
      type: string
    role:
      type: string
    joinedAt:
      type: string
      format: date-time
```

```
CreateItemRequest:
  type: object
  required:
    - type
    - data
  properties:
    type:
      type: string
      enum: [flight, hotel, activity, transfer]
      example: "hotel"
    data:
      type: object
      example: {"name": "Hotel Roma", "dates": "2024-07-15/2024-
07-20"}
    externalId:
      type: string
      description: ID из внешней системы (если есть)
      example: "booking_hotel_12345"
```

```
ItemResponse:
  type: object
  properties:
    id:
      type: string
      example: "item_789"
    type:
      type: string
    status:
      type: string
      enum: [proposed, confirmed, rejected, booked, paid]
      example: "proposed"
```

```
data:
  type: object
authorId:
  type: string
createdAt:
  type: string
  format: date-time
updatedAt:
  type: string
  format: date-time
version:
  type: integer
  example: 3
```

===== ДОБАВЛЕННЫЕ МОДЕЛИ (исправление ошибок) =====

```
UpdateItemRequest:
  type: object
  required:
    - data
    - version
  properties:
    data:
      type: object
      example: {"name": "Hotel Roma Updated", "dates": "2024-
07-15/2024-07-22"}
    version:
      type: integer
      description: Текущая версия элемента для оптимистичной
блокировки
      example: 3
```

```
BookingResponse:
  type: object
  properties:
    bookingId:
      type: string
      example: "booking_123"
    status:
      type: string
      enum: [pending, confirmed, failed]
      example: "pending"
    externalReference:
      type: string
      description: Идентификатор брони во внешней системе
      example: "ext_booking_abc123"
    estimatedCompletionTime:
      type: string
      format: date-time
      example: "2024-03-25T15:30:00Z"
```

```
HistoryEntry:
  type: object
```

```
properties:
  id:
    type: string
    example: "history_456"
  action:
    type: string
    enum: [create, update, delete, confirm, reject]
    example: "update"
  itemId:
    type: string
    example: "item_789"
  userId:
    type: string
    example: "user_123"
  timestamp:
    type: string
    format: date-time
    example: "2024-03-25T14:30:00Z"
  changes:
    type: object
    description: Измененные данные
```

```
ActivityOption:
  type: object
  properties:
    id:
      type: string
      example: "activity_123"
    name:
      type: string
      example: "Экскурсия в Колизей"
    category:
      type: string
      enum: [tour, show, sport, food, shopping]
      example: "tour"
    price:
      type: number
      example: 2500.50
    duration:
      type: integer
      description: Длительность в минутах
      example: 180
    rating:
      type: number
      example: 4.7
```

```
PackageOption:
  type: object
  properties:
    id:
      type: string
      example: "package_123"
    name:
```

```
    type: string
    example: "Тур в Италию на 7 дней"
  destination:
    type: string
    example: "Италия"
  duration:
    type: integer
    example: 7
  price:
    type: number
    example: 85000
  includes:
    type: array
    items:
      type: string
    example: ["перелет", "отель", "завтраки", "экскурсии"]
```

```
ServerEvent:
  type: object
  properties:
    eventId:
      type: string
      example: "server_event_789"
    action:
      type: string
      enum: [create, update, delete, confirm, reject]
    itemId:
      type: string
    data:
      type: object
    serverTimestamp:
      type: string
      format: date-time
    serverVersion:
      type: integer
      example: 47
```

```
FlightStatus:
  type: object
  properties:
    flightId:
      type: string
      example: "SU123"
    flightNumber:
      type: string
      example: "SU 123"
    status:
      type: string
      enum: [scheduled, boarding, delayed, cancelled, landed]
      example: "scheduled"
    departureGate:
      type: string
      example: "A12"
```

```
arrivalGate:
  type: string
  example: "B5"
scheduledDeparture:
  type: string
  format: date-time
estimatedDeparture:
  type: string
  format: date-time
scheduledArrival:
  type: string
  format: date-time
estimatedArrival:
  type: string
  format: date-time
```

```
BookingStatus:
  type: object
  properties:
    bookingId:
      type: string
      example: "booking_abc123"
    status:
      type: string
      enum: [confirmed, pending, cancelled, no_show]
      example: "confirmed"
    checkIn:
      type: string
      format: date
      example: "2024-07-15"
    checkOut:
      type: string
      format: date
      example: "2024-07-20"
    roomNumber:
      type: string
      example: "312"
```

```
PaymentResponse:
  type: object
  properties:
    paymentId:
      type: string
      example: "pay_12345"
    status:
      type: string
      enum: [pending, completed, failed, refunded]
      example: "pending"
    amount:
      type: number
      example: 15000.50
    currency:
      type: string
```



```
    example: "RUB"
  paymentUrl:
    type: string
    description: Ссылка для оплаты (для redirect)
    example: "https://payment.traveltech.com/pay/pay_12345"
```

```
PaymentHistoryEntry:
  type: object
  properties:
    paymentId:
      type: string
    itemId:
      type: string
    amount:
      type: number
    currency:
      type: string
    status:
      type: string
    method:
      type: string
    timestamp:
      type: string
      format: date-time
```

```
PassportData:
  type: object
  required:
    - passportNumber
    - lastName
    - firstName
    - birthDate
    - expiryDate
  properties:
    passportNumber:
      type: string
      example: "123456789"
    lastName:
      type: string
      example: "Иванов"
    firstName:
      type: string
      example: "Иван"
    middleName:
      type: string
      example: "Иванович"
    birthDate:
      type: string
      format: date
      example: "1990-01-15"
    expiryDate:
      type: string
      format: date
```

```
    example: "2030-01-15"
nationality:
  type: string
  example: "RU"
```

Document:

```
  type: object
  properties:
    id:
      type: string
      example: "doc_123"
    type:
      type: string
      enum: [ticket, voucher, receipt, insurance, visa]
      example: "ticket"
    name:
      type: string
      example: "Авиабилет Москва-Рим"
    fileUrl:
      type: string
```

example:

```
"https://storage.traveltech.com/documents/ticket_123.pdf"
  createdAt:
    type: string
    format: date-time
  expiresAt:
    type: string
    format: date-time
```

Модели для синхронизации

SyncRequest:

```
  type: object
  properties:
    lastSyncVersion:
      type: integer
      description: Последняя версия, известная клиенту
      example: 42
    clientEvents:
      type: array
      items:
        $ref: '#/components/schemas/ClientEvent'
```

ClientEvent:

```
  type: object
  properties:
    eventId:
      type: string
    action:
      type: string
      enum: [create, update, delete, confirm, reject]
    itemId:
      type: string
    data:
```

```
    type: object
  clientTimestamp:
    type: string
    format: date-time
```

```
SyncResponse:
  type: object
  properties:
    currentTripVersion:
      type: integer
      example: 50
    serverEvents:
      type: array
      items:
        $ref: '#/components/schemas/ServerEvent'
    conflicts:
      type: array
      items:
        $ref: '#/components/schemas/Conflict'
```

```
Conflict:
  type: object
  properties:
    conflictId:
      type: string
    description:
      type: string
      example: "Конфликт редактирования рейса SU-123"
    options:
      type: array
      items:
        type: object
```

```
ResolveConflictRequest:
  type: object
  required:
    - selectedOption
  properties:
    selectedOption:
      type: integer
      description: Индекс выбранного варианта
    comment:
      type: string
      description: Комментарий к разрешению конфликта
```

Модели для поиска

```
TransportOption:
  type: object
  properties:
    id:
      type: string
      example: "flight_su123"
  type:
```

```
    type: string
    example: "flight"
  carrier:
    type: string
    example: "Аэрофлот"
  departure:
    type: string
    format: date-time
  arrival:
    type: string
    format: date-time
  price:
    type: number
    example: 15000.50
```

```
HotelOption:
  type: object
  properties:
    id:
      type: string
      example: "hotel_123"
    name:
      type: string
      example: "Hotel Roma"
    price:
      type: number
      example: 12000.00
    rating:
      type: number
      example: 4.5
    address:
      type: string
      example: "Via del Corso, 123, Rome"
    amenities:
      type: array
      items:
        type: string
      example: ["WiFi", "Бассейн", "Завтрак"]
```

```
Recommendation:
  type: object
  properties:
    id:
      type: string
      example: "rec_123"
    type:
      type: string
      example: "hotel"
    title:
      type: string
      example: "Отель с видом на Колизей"
    matchScore:
      type: number
```

```
    example: 0.92
  reason:
    type: string
    example: "На основе вашего интереса к историческим местам"
  price:
    type: number
    example: 14500.00
```

```
PaymentRequest:
  type: object
  properties:
    amount:
      type: number
      example: 15000.50
    currency:
      type: string
      default: "RUB"
      example: "RUB"
    method:
      type: string
      enum: ["card", "apple_pay", "google_pay"]
      example: "card"
    cardToken:
      type: string
      description: Токен карты (если метод = card)
      example: "tok_123456789"
```

Тестирование YAML-кода на платформе SwaggerEditor.

TravelTech Trip Planning API 1.0.0 OAS 3.0

API для совместного планирования путешествий с поддержкой офлайн-работы и конфликт-менеджмента.

Trips Управление поездками

POST	/trips	Создать новую поездку	▼
GET	/trips	Получить список своих поездок	▼
GET	/trips/{tripId}	Получить полное текущее состояние поездки	▼
DELETE	/trips/{tripId}	Удалить поездку	▼
POST	/trips/{tripId}/members	Добавить участника в поездку	▼
GET	/trips/{tripId}/members	Получить список участников поездки	▼
DELETE	/trips/{tripId}/members/{userId}	Удалить участника из поездки	▼

Отображение Trips эндпоинтов

GET

/trips/{tripId}/members

Получить список участников поездки

^

Parameters

Try it out

Name	Description
tripId * required string (path)	<input type="text" value="tripId"/>

Responses

Code	Description	Links
200	Список участников поездки Media type <div>application/json</div> Controls Accept header. Example Value Schema <div> <pre>[{ "userId": "string", "role": "string", "joinedAt": "2026-01-29T17:44:45.733Z" }]</pre> </div>	No links

Развернутое описание /trips/{tripId}/members

Schemas

CreateTripRequest

name*

string

example: Отпуск в Италии, июль 2024

startDate*

string(\$date)

example: 2024-07-15

endDate

string(\$date)

example: 2024-07-25

description

string

example: Семейный отпуск в Риме и Флоренции

TripResponse

id

string

example: trip_123

name

string

status

string

example: planning

Enum:

[planning, active, completed, archived]

createdAt

string(\$date-time)

updatedAt

string(\$date-time)

items

[ItemResponse > {...}]

members

[MemberResponse > {...}]

AddMemberRequest

>

Компоненты, схемы (= структуры данных)

Примеры запросов и ответов API

Запрос: Создание поездки

POST /trips

Content-Type: application/json

Authorization: Bearer eyJhbGciOiJIUzI1NiIs...

```
{
  "name": "Отпуск в Италии, июль 2024",
  "ownerId": "user_123"
}
```

Ответ:

200 OK

Content-Type: application/json

```
{
  "id": "trip_abc123def456",
  "name": "Отпуск в Италии, июль 2024",
  "ownerId": "user_123",
  "version": 1,
  "createdAt": "2024-01-15T10:30:00Z",
  "status": "PLANNING",
  "members": ["user_123"],
  "items": []
}
```

Запрос: Добавление отеля

POST /trips/trip_abc123def456/items

Content-Type: application/json

Authorization: Bearer eyJhbGciOiJIUzI1NiIs...

```
{
  "type": "HOTEL",
  "data": {
    "name": "Hotel Bella Vista",
    "checkIn": "2024-07-15",
    "checkOut": "2024-07-22",
    "rooms": 2
  }
}
```

Ответ:

202 Accepted

Content-Type: application/json

```
{
  "id": "item_hotel_789xyz",
}
```

```

"type": "HOTEL",
"status": "PROPOSED",
"data": {
  "name": "Hotel Bella Vista",
  "checkIn": "2024-07-15",
  "checkOut": "2024-07-22",
  "rooms": 2
},
"authorId": "user_123",
"version": 1,
"createdAt": "2024-01-15T11:15:00Z"
}

```



Структура хранения нужной информации в базе данных

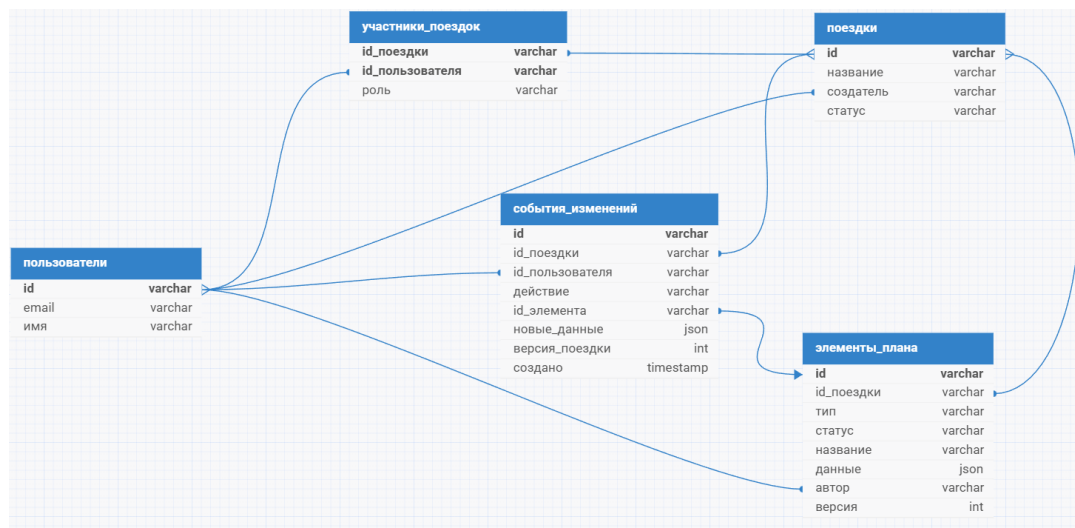


схема в case-системе dbdesigner

```

-- Создание таблицы пользователей
CREATE TABLE пользователи (
  id VARCHAR(36) PRIMARY KEY DEFAULT gen_random_uuid(),
  email VARCHAR(255) UNIQUE NOT NULL,
  имя VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Создание таблицы поездок
CREATE TABLE поездки (
  id VARCHAR(36) PRIMARY KEY DEFAULT gen_random_uuid(),
  название VARCHAR(500) NOT NULL,
  создатель VARCHAR(36) NOT NULL,
  статус VARCHAR(50) DEFAULT 'активна',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

```



```

        CONSTRAINT fk_поездки_создатель
            FOREIGN KEY (создатель)
            REFERENCES пользователи(id)
            ON DELETE CASCADE
    );

-- Создание таблицы участников поездок
CREATE TABLE участники_поездов (
    id_поездки VARCHAR(36) NOT NULL,
    id_пользователя VARCHAR(36) NOT NULL,
    роль VARCHAR(50) DEFAULT 'участник',
    joined_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    PRIMARY KEY (id_поездки, id_пользователя),

    CONSTRAINT fk_участники_поездка
        FOREIGN KEY (id_поездки)
        REFERENCES поездки(id)
        ON DELETE CASCADE,

    CONSTRAINT fk_участники_пользователь
        FOREIGN KEY (id_пользователя)
        REFERENCES пользователи(id)
        ON DELETE CASCADE
);

-- Создание таблицы элементов плана
CREATE TABLE элементы_плана (
    id VARCHAR(36) PRIMARY KEY DEFAULT gen_random_uuid(),
    id_поездки VARCHAR(36) NOT NULL,
    тип VARCHAR(50) NOT NULL,
    статус VARCHAR(50) DEFAULT 'предложен',
    название VARCHAR(500) NOT NULL,
    данные JSONB NOT NULL,
    автор VARCHAR(36) NOT NULL,
    версия INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT fk_элементы_поездка
        FOREIGN KEY (id_поездки)
        REFERENCES поездки(id)
        ON DELETE CASCADE,

    CONSTRAINT fk_элементы_автор
        FOREIGN KEY (автор)
        REFERENCES пользователи(id)
        ON DELETE CASCADE
);

-- Создание таблицы событий изменений
CREATE TABLE события_изменений (

```

```

id VARCHAR(36) PRIMARY KEY DEFAULT gen_random_uuid(),
id_поездки VARCHAR(36) NOT NULL,
id_пользователя VARCHAR(36) NOT NULL,
действие VARCHAR(50) NOT NULL,
id_элемента VARCHAR(36),
новые_данные JSONB,
версия_поездки INTEGER NOT NULL,
создано TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

CONSTRAINT fk_события_поездка
    FOREIGN KEY (id_поездки)
    REFERENCES поездки(id)
    ON DELETE CASCADE,

CONSTRAINT fk_события_пользователь
    FOREIGN KEY (id_пользователя)
    REFERENCES пользователи(id)
    ON DELETE CASCADE,

CONSTRAINT fk_события_элемент
    FOREIGN KEY (id_элемента)
    REFERENCES элементы_плана(id)
    ON DELETE SET NULL
);

-- Создание индексов для производительности
CREATE INDEX idx_поездки_создатель ON поездки(создатель);
CREATE INDEX idx_участники_пользователь ON
участники_поездов(id_пользователя);
CREATE INDEX idx_элементы_поездка ON элементы_плана(id_поездки);
CREATE INDEX idx_элементы_автор ON элементы_плана(автор);
CREATE INDEX idx_элементы_статус ON элементы_плана(статус);
CREATE INDEX idx_события_поездка ON события_изменений(id_поездки);
CREATE INDEX idx_события_пользователь ON
события_изменений(id_пользователя);
CREATE INDEX idx_события_элемент ON
события_изменений(id_элемента);
CREATE INDEX idx_события_создано ON события_изменений(создано
DESC);

```

Связи между таблицами в базе данных TravelTech:

- 1. один ко многим (пользователи → поездки)**
пользователи.id → поездки.создатель
Один пользователь может создать много поездов
- 2. один ко многим (поездки → участники_поездов)**
поездки.id → участники_поездов.id_поездки
Одна поездка может иметь много участников

3. **один ко многим (пользователи → участники_поездов)**
пользователи.id → участники_поездов.id_пользователя
Один пользователь может участвовать во многих поездках
4. **один ко многим (поездки → элементы_плана)**
поездки.id → элементы_плана.id_поездки
Одна поездка содержит много элементов плана
5. **один ко многим (пользователи → элементы_плана)**
пользователи.id → элементы_плана.автор
Один пользователь может создать много элементов плана
6. **один ко многим (поездки → события_изменений)**
поездки.id → события_изменений.id_поездки
Одна поездка имеет много событий изменений
7. **один ко многим (пользователи → события_изменений)**
пользователи.id → события_изменений.id_пользователя
Один пользователь совершает много действий по изменению плана
8. **один ко многим (элементы_плана → события_изменений)**
элементы_плана.id → события_изменений.id_элемента
Один элемент плана может иметь много связанных событий изменений

Вставляем данные:

Сценарий 1: Создание пользователей и поездки

```
-- 1. Создаем пользователей
INSERT INTO пользователи (id, email, имя) VALUES
('usr_001', 'ivan.petrov@example.com', 'Иван Петров'),
('usr_002', 'anna.sidorova@example.com', 'Анна Сидорова'),
('usr_003', 'sergey.ivanov@example.com', 'Сергей Иванов'),
('usr_004', 'maria.kuznetsova@example.com', 'Мария Кузнецова');

-- 2. Иван создает поездку
INSERT INTO поездки (id, название, создатель, статус,
версия_поездки) VALUES
('trip_001', 'Семейный отдых в Турции', 'usr_001', 'активна', 0);

-- 3. Добавляем участников поездки
INSERT INTO участники_поездов (id_поездки, id_пользователя, роль)
VALUES
('trip_001', 'usr_001', 'организатор'),
('trip_001', 'usr_002', 'участник'),
('trip_001', 'usr_003', 'участник'),
('trip_001', 'usr_004', 'участник');
```

таблица	id	имя	email	созданных_поездов
Поездка	trip_001	Семейный отдых в Турции	Создатель: Иван Петров	3
Пользователи	usr_001	Иван Петров	ivan.petrov@example.com	1
Пользователи	usr_002	Анна Сидорова	anna.sidorova@example.com	0
Пользователи	usr_003	Сергей Иванов	sergey.ivanov@example.com	0

Сценарий 2: Один участник добавляет рейс, отель и активность

```
-- 1. Иван добавляет рейс
INSERT INTO элементы_плана (id, id_поездки, тип, статус, название,
данные, автор) VALUES
('item_001', 'trip_001', 'рейс', 'предложен', 'Аэрофлот SU-2134',
'{"рейс": "SU-2134", "откуда": "Москва (Шереметьево)", "куда":
"Анталья",
"дата": "2024-06-15", "время": "08:30", "цена": 45000,
"авиакомпания": "Аэрофлот", "пассажиры": 4}'::jsonb,
'usr_001');

-- 2. Фиксируем событие
INSERT INTO события_изменений (id, id_поездки, id_пользователя,
действие, id_элемента, новые_данные, версия_поездки) VALUES
('evt_001', 'trip_001', 'usr_001', 'создать', 'item_001',
'{"тип": "рейс", "название": "Аэрофлот SU-2134"}'::jsonb, 1);

-- 3. Иван добавляет отель
INSERT INTO элементы_плана (id, id_поездки, тип, статус, название,
данные, автор) VALUES
('item_002', 'trip_001', 'отель', 'предложен', 'Rixos Premium
Belek',
'{"название_отеля": "Rixos Premium Belek", "категория": "5
звезд",
"дата_заезда": "2024-06-15", "дата_выезда": "2024-06-25",
"ночей": 10, "цена_за_ночь": 12000, "тип_номера": "Deluxe
Room"}'::jsonb,
'usr_001');

-- 4. Иван добавляет активность
INSERT INTO элементы_плана (id, id_поездки, тип, статус, название,
данные, автор) VALUES
('item_003', 'trip_001', 'активность', 'предложен', 'Экскурсия в
Памуккале',
'{"название": "Экскурсия в Памуккале", "дата": "2024-06-18",
"продолжительность": "10 часов", "цена": 3500,
"тип": "групповая экскурсия"}'::jsonb,
'usr_001');

-- 5. Обновляем версию поездки
UPDATE поездки SET версия_поездки = версия_поездки + 3 WHERE id =
'trip_001';
```

раздел	id	тип	название	статус	автор	версия	время_добавления
Элементы плана	item_001	рейс	Аэрофлот SU-2134	предложен	Иван Петров	0	18:12
Элементы плана	item_002	отель	Rixos Premium Belek	предложен	Иван Петров	0	18:12
Элементы плана	item_003	активность	Экскурсия в Памуккале	предложен	Иван Петров	0	18:12

Сценарий 3: Другой участник предлагает альтернативные варианты

```
-- 1. Анна предлагает альтернативный рейс
```

```
INSERT INTO элементы_плана (id, id_поездки, тип, статус, название,
данные, автор) VALUES
('item_004', 'trip_001', 'рейс', 'предложен', 'Turkish Airlines
TK-415',
  '{"рейс": "TK-415", "откуда": "Москва (Внуково)", "куда":
"Анталья",
  "дата": "2024-06-15", "время": "10:45", "цена": 42000,
  "авиакомпания": "Turkish Airlines"}':::jsonb,
'usr_002');

-- 2. Анна предлагает альтернативный отель
INSERT INTO элементы_плана (id, id_поездки, тип, статус, название,
данные, автор) VALUES
('item_005', 'trip_001', 'отель', 'предложен', 'Maxx Royal Belek
Golf Resort',
  '{"название_отеля": "Maxx Royal Belek Golf Resort", "категория":
"5 звезд",
  "дата_заезда": "2024-06-15", "дата_выезда": "2024-06-25",
  "ночей": 10, "цена_за_ночь": 15000, "тип_номера": "Family
Room"}':::jsonb,
'usr_002');

-- 3. Фиксируем события
INSERT INTO события_изменений (id, id_поездки, id_пользователя,
действие, id_элемента, новые_данные, версия_поездки) VALUES
('evt_002', 'trip_001', 'usr_002', 'создать', 'item_004',
  '{"тип": "рейс", "название": "Turkish Airlines TK-415"}':::jsonb,
4),
('evt_003', 'trip_001', 'usr_002', 'создать', 'item_005',
  '{"тип": "отель", "название": "Maxx Royal Belek"}':::jsonb, 5);

-- 4. Обновляем версию поездки
UPDATE поездки SET версия_поездки = версия_поездки + 2 WHERE id =
'trip_001';
```

тип	всего_предложений	авторы	варианты
активность	1	Иван Петров	Экскурсия в Памуккале
отель	2	Иван Петров, Анна Сидорова	Rixos Premium Belek Maxx Royal Belek Golf Resor
рейс	2	Иван Петров, Анна Сидорова	Аэрофлот SU-2134 Turkish Airlines TK-415

Сценарий 4: Кто-то редактирует элемент

```
-- 1. Иван редактирует рейс (меняет время)
UPDATE элементы_плана
SET данные = данные || '{"время": "09:15", "комментарий": "Более
удобное время", "цена": 45500}':::jsonb,
  версия = версия + 1,
  updated_at = CURRENT_TIMESTAMP
WHERE id = 'item_001'
AND автор = 'usr_001';
```

```
-- 2. Фиксируем событие редактирования
INSERT INTO события_изменений (id, id_поездки, id_пользователя,
действие, id_элемента, новые_данные, версия_поездки) VALUES
('evt_004', 'trip_001', 'usr_001', 'обновить', 'item_001',
'{"изменения": {"время": "08:30 → 09:15", "цена": "45000 →
45500"}, "комментарий": "Более удобное время"}'::jsonb, 6);

-- 3. Обновляем версию поездки
UPDATE поездки SET версия_поездки = версия_поездки + 1 WHERE id =
'trip_001';
```

запрос	id	название	новое_время	новая_цена	комментарий	версия	обновлено
История изменений рейса	item_001	Аэрофлот SU-2134	09:15	45500	Более удобное время	1	18:15:13

Сценарий 5: Кто-то удаляет элемент

```
-- 1. Анна удаляет свой альтернативный рейс
DELETE FROM элементы_плана
WHERE id = 'item_004'
AND автор = 'usr_002';

-- 2. Фиксируем событие удаления
INSERT INTO события_изменений (id, id_поездки, id_пользователя,
действие, id_элемента, новые_данные, версия_поездки) VALUES
('evt_005', 'trip_001', 'usr_002', 'удалить', NULL,
'{"удаленный_элемент": "item_004", "тип": "рейс", "название":
"Turkish Airlines TK-415", "причина": "выбран другой
вариант"}'::jsonb, 7);

-- 3. Обновляем версию поездки
UPDATE поездки SET версия_поездки = версия_поездки + 1 WHERE id =
'trip_001';
```

группа	тип	название	автор	элемент_статус
Удаленные (в событиях)	рейс	Turkish Airlines TK-415	Анна Сидорова	удален
Текущие элементы	активность	Экскурсия в Памуккале	Иван Петров	предложен
Текущие элементы	отель	Rixos Premium Belek	Иван Петров	предложен
Текущие элементы	отель	Maxx Royal Belek Golf Resort	Анна Сидорова	предложен

Сценарий 6: Подтверждение элемента

```
-- 1. Иван подтверждает отель
UPDATE элементы_плана
SET статус = 'подтвержден',
    версия = версия + 1,
    updated_at = CURRENT_TIMESTAMP
WHERE id = 'item_002'
AND id_поездки = 'trip_001';

-- 2. Фиксируем событие подтверждения
INSERT INTO события_изменений (id, id_поездки, id_пользователя,
действие, id_элемента, новые_данные, версия_поездки) VALUES
```

```
('evt_006', 'trip_001', 'usr_001', 'подтвердить', 'item_002',  
  '{"старый_статус": "предложен", "новый_статус":  
  "подтвержден"}'::jsonb, 8);
```

-- 3. Обновляем версию поездки

```
UPDATE поездки SET версия_поездки = версия_поездки + 1 WHERE id =  
'trip_001';
```

тип	статус	название	автор	индикатор	обновлено
отель	подтвержден	Rixos Premium Belek	Иван Петров	✓	18:19
активность	предложен	Экскурсия в Памуккале	Иван Петров	●	18:19
отель	предложен	Maxx Royal Belek Golf Resort	Анна Сидорова	●	18:19
рейс	предложен	Аэрофлот SU-2134	Иван Петров	●	18:19

Система обеспечивает синхронизированное планирование путешествий для групп с поддержкой офлайн-работы через механизм версионирования и разрешения конфликтов. REST API и оптимизированная база данных позволяют участникам совместно редактировать маршруты в реальном времени.