Техническая документация на разработку приложения Travel Booking

Оглавление

Глоссарий

Понятие (термин)	Определение
Бронь	Запись о заказе пользователя. Пример – бронь №56782 на авиарейс SU153
Турпакет	Сочетание билета, отеля и экскурсии. Пример – пакет «Париж на выходные»
Payment ID	Уникальный идентификатор транзакции. Пример – pmt_002913ab

Продуктовое описание

Реализовать функциональность для бронирования билета в приложении TravelBooking.

Основные акторы:

- Пользователь – конечный клиент сервиса. Пример действий - бронирование тура
- Администратор – управление контентом и пользователями. Пример действий – Добавление / удаление карточек отелей
- Система оплаты – внешний сервис. Проверка и прием платежей

Требования

Бизнес-требование

Предоставить возможность удобного поиска среди существующих предложений авиакомпаний и покупки билетов на одном сайте.

Бизнес-правило

Стоимость билетов обновляется в режиме реального времени. Финальная стоимость определяется авиакомпанией-поставщиком услуги.

Пользовательское

Клиент имеет возможность в режиме реального времени выбирать среди предложений различных авиакомпаний, бронировать и покупать билеты.

Перечень функций (функциональные требования):

- Поиск и бронирование авиабилетов, отелей и экскурсий
- Управление своими бронями в личном кабинете
- Получение уведомлений о статусах бронирования

Нефункциональные требования:
Реализовано как веб-приложение для ОС mac, windows, linux

Формализация требований

Пользовательские истории User story

Пользовательские истории для актора “Пользователь”

Поиск и бронирование услуг

US-001: Поиск авиабилетов
Как пользователь,
я хочу искать авиабилеты по направлениям и датам,
чтобы найти подходящие варианты для путешествия.

US-002: Поиск отелей
Как пользователь,
я хочу искать отели по местоположению и датам заезда/выезда,
чтобы найти комфортное жилье для отдыха.

US-003: Поиск экскурсий
Как пользователь,
я хочу искать экскурсии по городам и датам,
чтобы спланировать культурную программу путешествия.

US-004: Бронирование авиабилета
Как пользователь,
я хочу бронировать выбранный авиабилет с указанием пассажиров,
чтобы зафиксировать место на рейсе.

US-005: Бронирование отеля
Как пользователь,
я хочу бронировать выбранный номер в отеле,
чтобы обеспечить себе проживание во время поездки.

US-006: Бронирование экскурсии
Как пользователь,
я хочу бронировать выбранную экскурсию,
чтобы гарантировать участие в мероприятии.

US-007: Формирование турпакета
Как пользователь,
я хочу создавать турпакеты, объединяя билеты, отели и экскурсии,
чтобы получить комплексное туристическое предложение со скидкой.

Управление бронями

US-008: Просмотр списка броней
Как пользователь,
я хочу видеть все свои брони в личном кабинете,
чтобы отслеживать текущие и завершенные поездки.

US-009: Просмотр деталей брони
Как пользователь,
я хочу видеть подробную информацию о конкретной брони,

чтобы знать все детали заказа.

US-010: Отмена бронирования
Как пользователь,
я хочу отменять бронирования в соответствии с условиями отмены,
чтобы освободить место и получить возврат средств при возможности.

US-011: Изменение бронирования
Как пользователь,
я хочу изменять параметры существующей брони,
чтобы скорректировать поездку под изменившиеся обстоятельства.

Уведомления и оплата

US-012: Получение уведомлений о статусе брони
Как пользователь,
я хочу получать уведомления об изменениях статуса бронирования,
чтобы быть в курсе состояния своего заказа.

US-013: Оплата бронирования
Как пользователь,
я хочу оплачивать бронирования через интегрированную систему оплаты,
чтобы завершить процесс оформления заказа.

US-014: Получение подтверждения оплаты
Как пользователь,
я хочу получать подтверждение об успешной оплате с Payment ID,
чтобы иметь доказательство транзакции.

US-015: Получение напоминаний о поездке
Как пользователь,
я хочу получать напоминания о предстоящих поездках,
чтобы не забыть о запланированном путешествии.

Пользовательские истории для актора «Администратор»

Управление контентом

US-016: Добавление карточек отелей
Как администратор,
я хочу добавлять новые карточки отелей в систему,
чтобы расширять базу доступных для бронирования объектов.

US-017: Редактирование карточек отелей
Как администратор,
я хочу редактировать информацию об отелях,
чтобы поддерживать актуальность данных в системе.

US-018: Удаление карточек отелей
Как администратор,
я хочу удалять карточки отелей из системы,
чтобы убирать недоступные или неактуальные объекты.

US-019: Управление авиарейсами
Как администратор,
я хочу добавлять, редактировать и удалять информацию об авиарейсах,
чтобы поддерживать актуальное расписание.

US-020: Управление экскурсиями
Как администратор,
я хочу добавлять, редактировать и удалять информацию об экскурсиях,
чтобы обновлять культурную программу для пользователей.

US-021: Создание турпакетов
Как администратор,
я хочу создавать готовые турпакеты из комбинаций услуг,
чтобы предлагать пользователям комплексные решения.

Управление пользователями

US-022: Просмотр списка пользователей
Как администратор,
я хочу видеть список всех зарегистрированных пользователей,
чтобы управлять пользовательской базой.

US-023: Блокировка пользователей
Как администратор,
я хочу блокировать пользователей за нарушения правил,
чтобы поддерживать порядок в системе.

US-024: Просмотр статистики бронирований
Как администратор,
я хочу видеть статистику по бронированиям,
чтобы анализировать популярность услуг и планировать развитие.

Управление бронированиями

US-025: Просмотр всех бронирований
Как администратор,
я хочу видеть все бронирования в системе,
чтобы контролировать текущие заказы.

US-026: Изменение статусов бронирований
Как администратор,
я хочу изменять статусы бронирований,
чтобы управлять процессом обработки заказов.

US-027: Отмена бронирований администратором
Как администратор,
я хочу отменять бронирования по техническим причинам,
чтобы решать проблемные ситуации.

Пользовательские истории для актора “Система оплаты”

US-028: Проверка платежных данных
Как система оплаты,
я хочу проверять корректность платежных данных пользователя,
чтобы обеспечить безопасность транзакций.

US-029: Обработка платежей
Как система оплаты,
я хочу обрабатывать платежи за бронирования,
чтобы завершать финансовые операции.

US-030: Генерация Payment ID
Как система оплаты,
я хочу генерировать уникальные идентификаторы транзакций,
чтобы обеспечить трассируемость платежей.

US-031: Возврат средств
Как система оплаты,
я хочу обрабатывать возвраты средств при отмене бронирований,
чтобы соблюдать условия отмены.

US-032: Уведомление о статусе платежа
Как система оплаты,
я хочу отправлять уведомления о статусе платежа в основную систему,
чтобы синхронизировать финансовые операции с состоянием бронирований.

Сводная таблица User Stories

ID	Актор	Краткое описание	Приоритет
US-001	Пользователь	Поиск авиабилетов	Высокий
US-002	Пользователь	Поиск отелей	Высокий
US-003	Пользователь	Поиск экскурсий	Высокий
US-004	Пользователь	Бронирование авиабилета	Высокий
US-005	Пользователь	Бронирование отеля	Высокий
US-006	Пользователь	Бронирование экскурсии	Высокий
US-007	Пользователь	Формирование турпакета	Средний
US-008	Пользователь	Просмотр списка броней	Высокий
US-009	Пользователь	Просмотр деталей брони	Высокий
US-010	Пользователь	Отмена бронирования	Средний
US-011	Пользователь	Изменение бронирования	Средний
US-012	Пользователь	Получение уведомлений	Средний
US-013	Пользователь	Оплата бронирования	Высокий

US-014	Пользователь	Получение подтверждения оплаты	Средний
US-015	Пользователь	Получение напоминаний	Низкий
US-016	Администратор	Добавление карточек отелей	Высокий
US-017	Администратор	Редактирование карточек отелей	Высокий
US-018	Администратор	Удаление карточек отелей	Высокий
US-019	Администратор	Управление авиарейсами	Высокий
US-020	Администратор	Управление экскурсиями	Высокий
US-021	Администратор	Создание турпакетов	Средний
US-022	Администратор	Просмотр списка пользователей	Средний
US-023	Администратор	Блокировка пользователей	Низкий
US-024	Администратор	Просмотр статистики	Средний
US-025	Администратор	Просмотр всех бронирований	Высокий
US-026	Администратор	Изменение статусов бронирований	Высокий
US-027	Администратор	Отмена бронирования	Средний
US-028	Система оплаты	Проверка платежных данных	Высокий
US-029	Система оплаты	Обработка платежей	Высокий
US-030	Система оплаты	Генерация Payment ID	Высокий
US-031	Система оплаты	Возврат средств	Средний
US-032	Система оплаты	Уведомление о статусе платежа	Высокий

Сценарий использования Use case

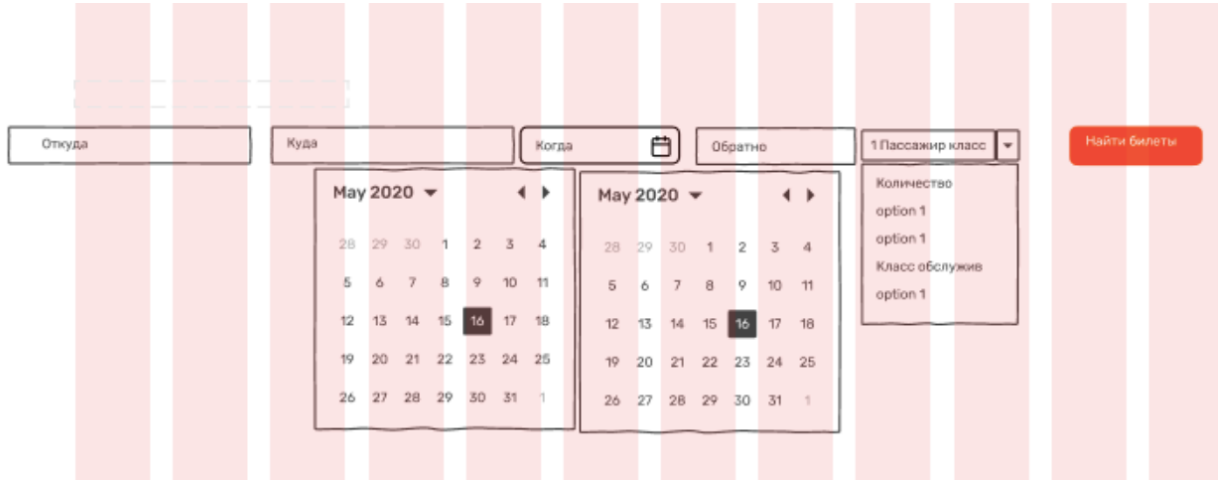
Название	UC-001. Бронирование авиабилета пользователем
User Story	US-004: Как пользователь, я хочу забронировать выбранный авиабилет с указанием пассажиров, чтобы зафиксировать место на рейсе
Акторы	Пользователь, система
Предусловия	Пользователь авторизован. База данных доступна для записи. Пользователь выполнил поиск и выбрал подходящий рейс
Ограничения	Бронирование доступно только для рейсов с доступными местами. Время на завершение бронирования: 15 минут.

	Максимум 9 пассажиров в одном бронировании. Оплата должна быть завершена в течение 15 минут после бронирования
Триггер	Пользователь нажимает кнопку “Оформить” на странице выбранного рейса
Основной сценарий	1. Пользователь выбирает рейс. 2. Система проверяет доступность мест на выбранном рейсе и отображает данные о билете. 3. Пользователь нажимает кнопку «Купить». 4. Система валидирует данные: проверяет актуальность цены, подтверждает наличие мест получает booking session подготавливает данные от партнера. 5. Система отображает форму ввода данных пассажиров и форму для выбора дополнительных услуг. 6. Пользователь заполняет данные пассажиров (ФИО, дата рождения, контактные данные, паспортные данные) 7. Пользователь выбирает дополнительные услуги. 8. Пользователь нажимает кнопку «Перейти на страницу оплаты».
Альтернативные сценарии	3а Места закончились - Если места на рейсе закончились – показать сообщение «Места закончились» и предложить альтернативные варианты 8а Ошибка валидации данных - Если данные пассажиров некорректны – подсветить поля с ошибками и показать сообщения 8б Ошибка резервирования - Если резервирование не удалось – показать сообщение «Ошибка бронирования» и предложить повторить
Исключительный сценарий	3б Отмена бронирования - Если пользователь отменяет бронирование – вернуть на страницу поиска рейсов
Результат/критерий успеха	Система забронировала билет, отведено время на ожидание оплаты от пользователя

Реализация для Frontend

Макеты

ID	Актор	Краткое описание	Приоритет
US-001	Пользователь	Поиск авиабилетов	Высокий

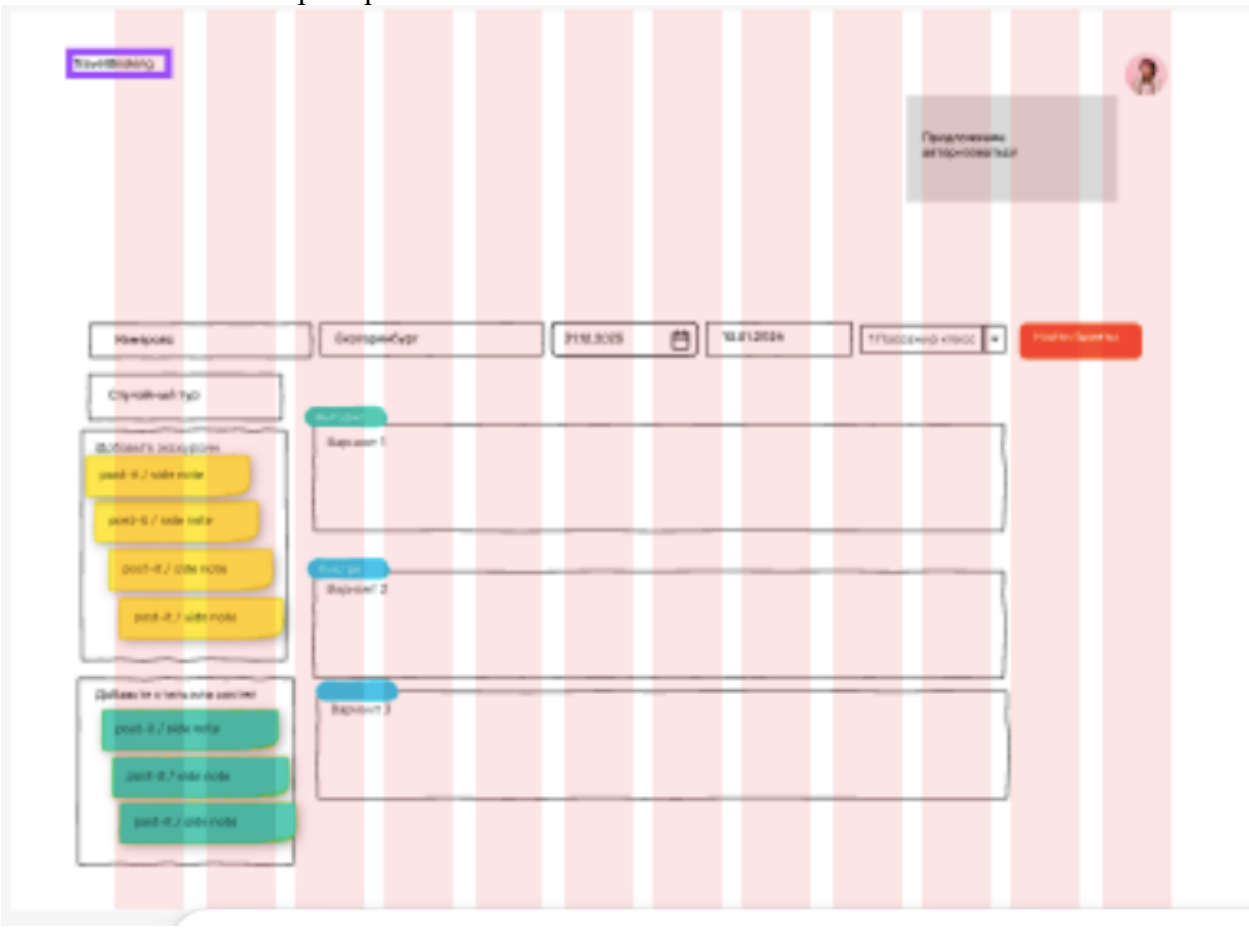


Тип	Наименование	Характеристика
Поле	«Откуда» «Куда»	Type: Autocomplete input Data type: string Source: cities API Min length: 2 Max length: 64 Required: yes Validation: - only letters - no digits - trim spaces Error: - «Введите город вылета»
Поле	«Когда» «Обратно»	Type: Date picker Data type: ISO date (YYYY-MM-DD) Min date: today Max date: today + 365 days Required: yes Timezone: local
Поле	«Пассажиры»	Type: Stepper / Select

		Data type: integer Range: 1–9 Default: 1
Кнопка	«Найти билеты»	Type: Primary button Action: submit search form State: - disabled if required fields empty - loading after click
Всплывающее окно	Ошибка: не заполнены обязательные поля	Error code: Empty_field Message: «Заполните обязательные поля»

ID	Актор	Краткое описание	Приоритет
US-004	Пользователь	Бронирование авиабилета	Высокий

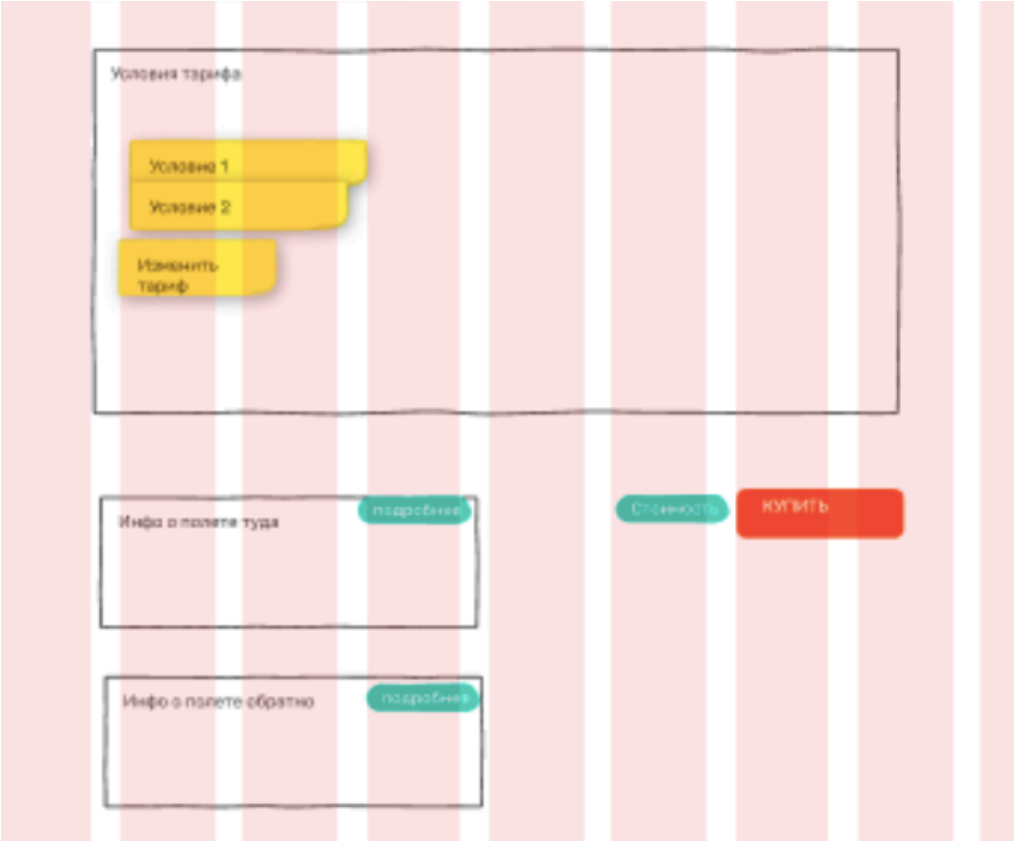
1. Пользователь выбирает рейс.



Тип	Наименование	Характеристика
Поле кликабельное	Карточка билета	Component: FlightCard Data: - flight_id: string

		- city_out: string - city_in: string - price: number - duration: number - stops: integer - airline_code: string State: - default - cheapest - fastest - no_seats
Поле	Цена	Type: currency Format: 12 345 ₽ Decimals: no
Всплывающее окно	Окно авторизации	Аутентификация - OAuth 2.0

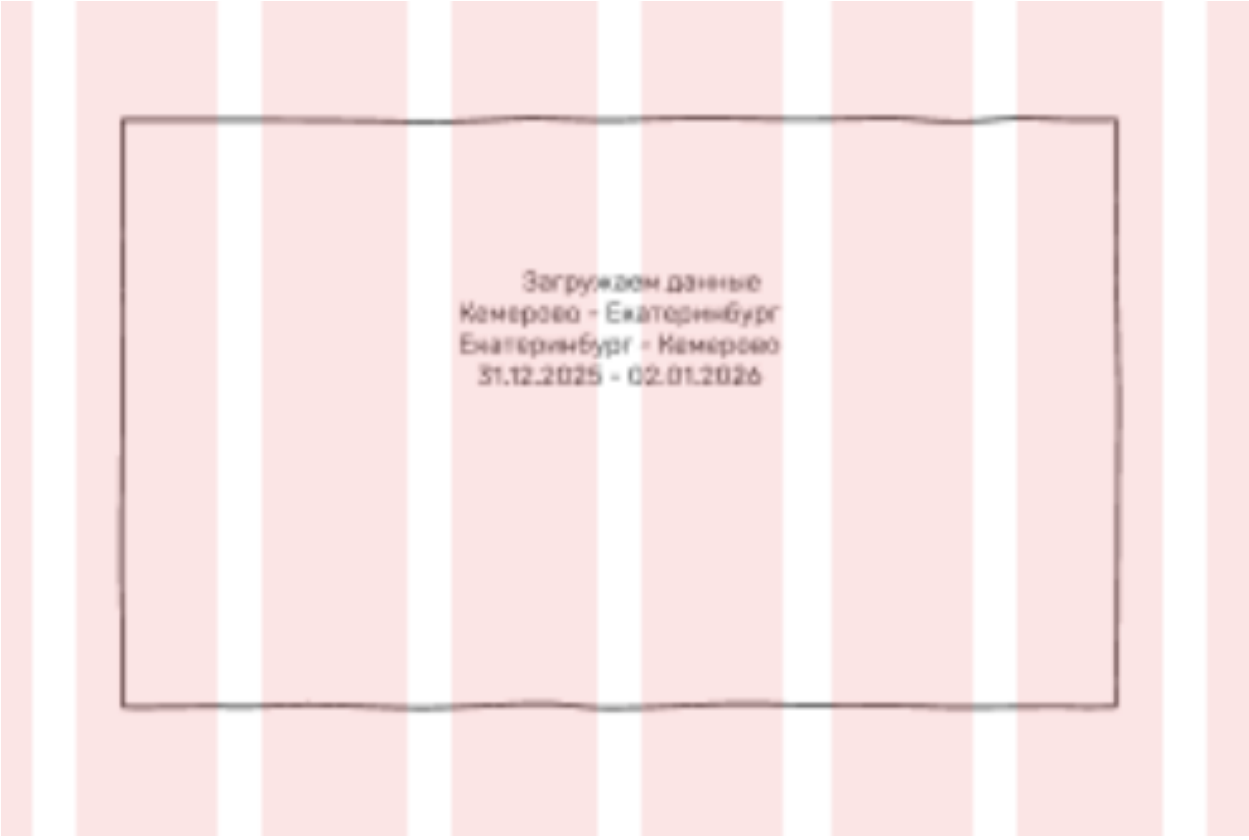
2. Система проверяет доступность мест на выбранном рейсе и отображает данные о билете.
3. Пользователь нажимает кнопку «Купить».



Тип	Наименование	Характеристика
		Component: SearchSummary Fields: - origin - destination

		- departure_date - return_date - passengers - class
Кнопка	«Купить»	Action: POST /booking-sessions Precondition: - flight.available = true
Поле	«Стоимость»	Type: currency Format: 12 345 ₽ Decimals: no
Всплывающее окно	Ошибка отсутствия мест	Error code: NO_SEATS Message: «Места закончились» Action: show alternatives
Всплывающее окно	Ошибка резервирования	Error code: BOOKING_FAILED Retry: allowed

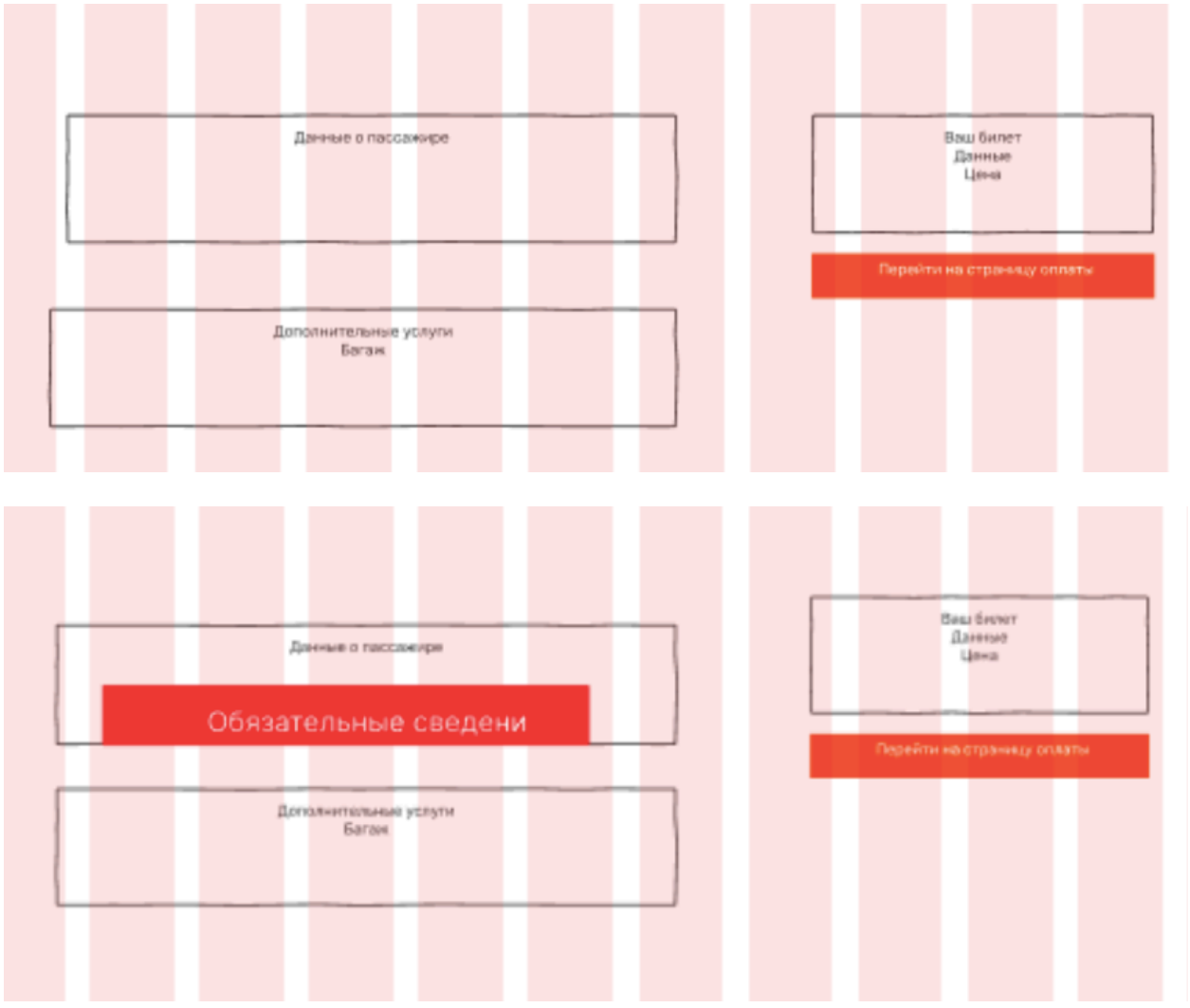
4. Система валидирует данные:
проверяет актуальность цены,
подтверждает наличие мест
получает booking session
подготавливает данные от партнера.



Тип	Наименование	Характеристика
Переходный экран (pre-booking)	Загружаем данные Туда – Обратно	UI type: Fullscreen modal / blocking overlay Z-index: max Scroll: disabled

	Дата туда – Дата обратно	Closable: no

5. Система отображает форму ввода данных пассажиров и форму для выбора дополнительных услуг.
6. Пользователь заполняет данные пассажиров (ФИО, дата рождения, контактные данные, паспортные данные)
7. Пользователь выбирает дополнительные услуги.
8. Пользователь нажимает кнопку «Перейти на страницу оплаты».

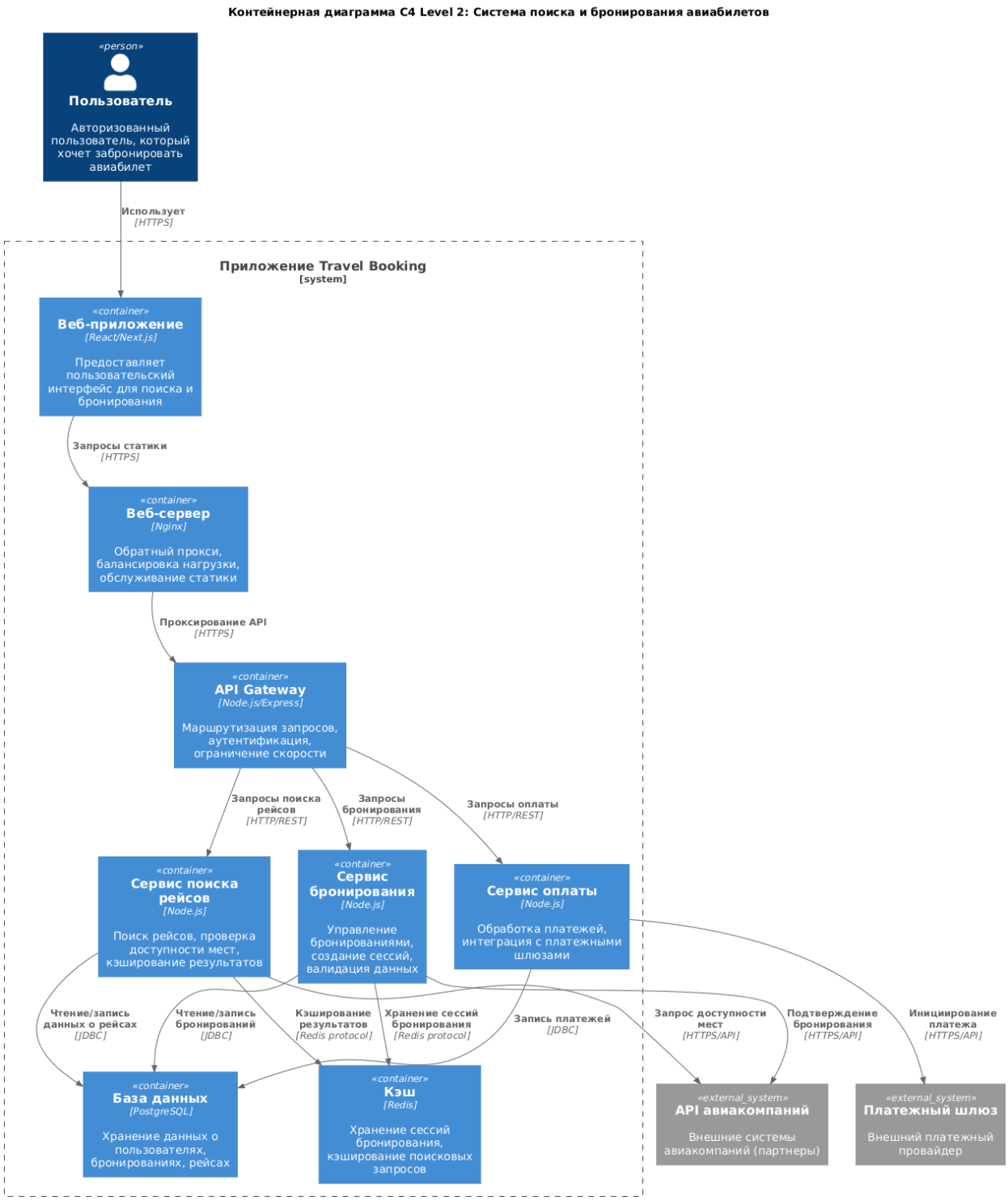


Тип	Наименование	Характеристика
Поле для ввода	Данные о пассажире: Гражданство Документ Пол	Type: Text input Data type: string Alphabet: latin only Min length: 2 Max length: 50 Required: yes

Поле для ввода	Данные о паспорте	Type: Text input Data type: string Pattern: /^[A-Z0-9]{6,9}$/ Required: yes
Кнопка	«Купить»	Action: POST /booking-sessions Precondition: - flight.available = true
Поле	«Стоимость»	Type: currency Format: 12 345 ₽ Decimals: no
Всплывающее окно	Не заполнены обязательные поля	Error code: Empty_field Message: «Заполните обязательные поля»

Реализация для Backend

Архитектура (C4)



Описание

Пользователь использует веб-приложением, которое через API Gateway инициирует поиск, бронирование и оплату авиабилета. Внутренние сервисы разделены по функциональности.

- Веб-сервер (Nginx):
    - Принимает HTTPS-запросы от веб-приложения

    - Маршрутизирует запросы
- Веб-приложение (React / Next.js) – предоставляет пользовательский интерфейс для поиска и бронирования билетов
    - Отправляет пользовательские запросы через REST API
    - Получает от API Gateway список доступных рейсов, данные о бронировании
- API Gateway (Node.js) – является единой точкой входа для всех клиентских запросов и маршрутизирует их к соответствующим внутренним сервисам
    - Принимает REST-запросы от веб-приложения
    - Выполняет проверку авторизации, валидацию запросов
    - Перенаправляет запросы к сервису поиска рейсов, к сервису бронирования, к сервису оплаты
- Сервис поиска рейсов (Node.js) – отвечает за поиск, фильтрацию и проверку доступности авиабилетов
- Сервис бронирования (Node.js) – центральный сервис бизнес-логики бронирования
    - Получает запрос от API Gateway на создание бронирования
    - Проверяет корректность данных пассажиров, доступность мест
    - Создает бронирование со статусом PENDING_PAYMENT
    - Сохраняет данные в PostgreSQL
    - Записывает сессию бронирования в Redis (TTL 15 минут)
    - Возвращает клиенту идентификатор бронирования и время истечения
- Сервис оплаты (Node.js)
    - Получает запрос от API Gateway на оплату бронирования
- База данных (PostgreSQL) – основное хранилище данных системы
- Кэш (Redis) – обеспечивает быстрый доступ к временным данным
- Внешние системы
    - API авиакомпаний
    - Платежный шлюз

Сервис бронирования

Сервис поиска рейсов управляет жизненным циклом брони: отвечает за создание бронирования и валидацию данных пассажиров.

Сервис	Delivery

Текстовый URL	https://travel-booking-test.ru
Описание	Микросервис обеспечивает бронирование авиабилетов и валидацию данных пассажиров.
База данных	Ссылка на раздел с БД
Методы	Ссылка на раздел с API

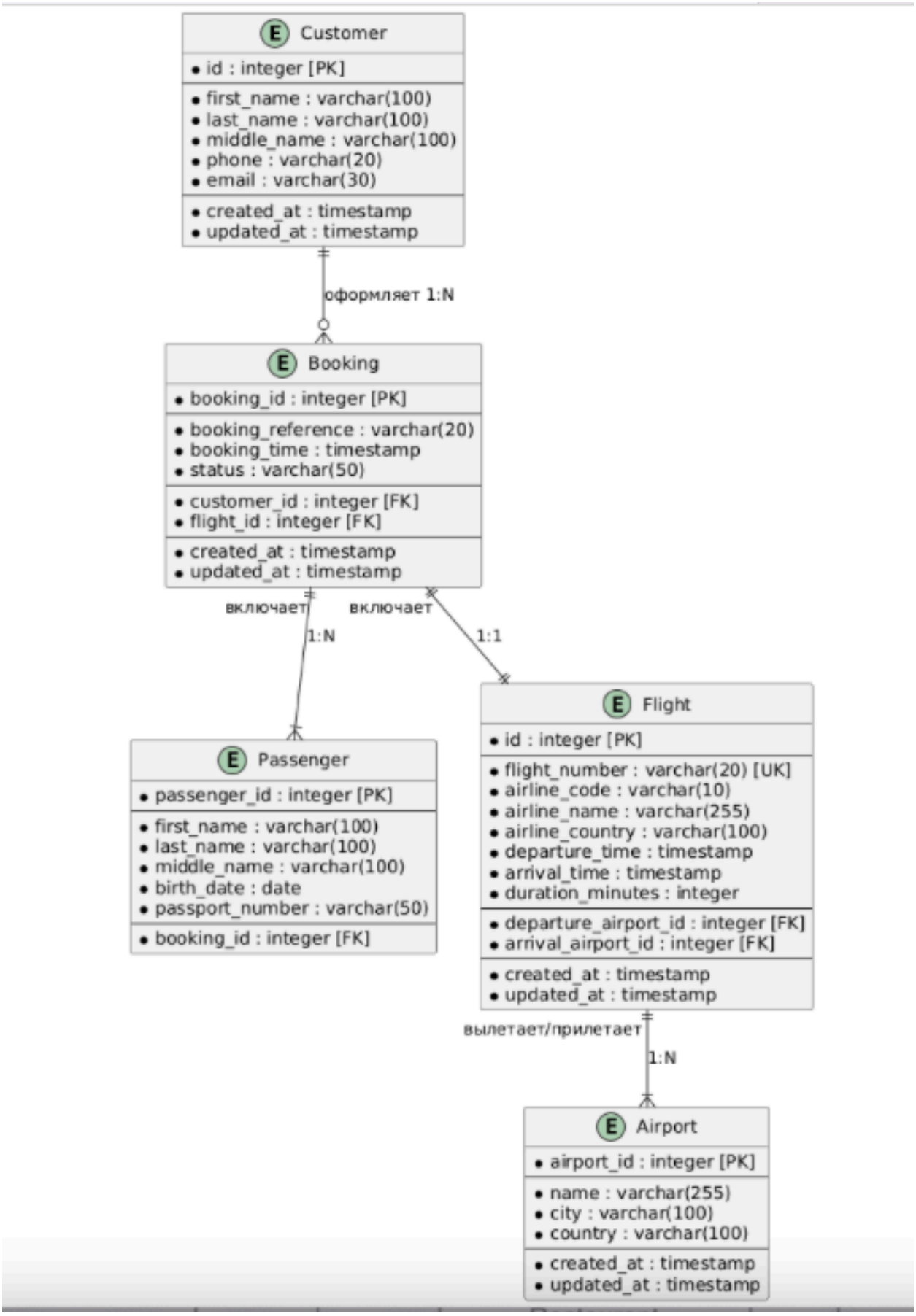
Алгоритм работы:

1. Получает запрос от API Gateway на создание бронирования после того, как пользователь нажал «Купить»
2. Проверяет:
   - Корректность данных пассажиров
   - Доступность мест
3. Создает бронирование со статусом PENDING_PAYMENT
4. Сохраняет данные в PostgreSQL
5. Записывает сессию бронирования в Redis (TTL 15 минут)
6. Возвращает клиенту идентификатор бронирования и время истечения

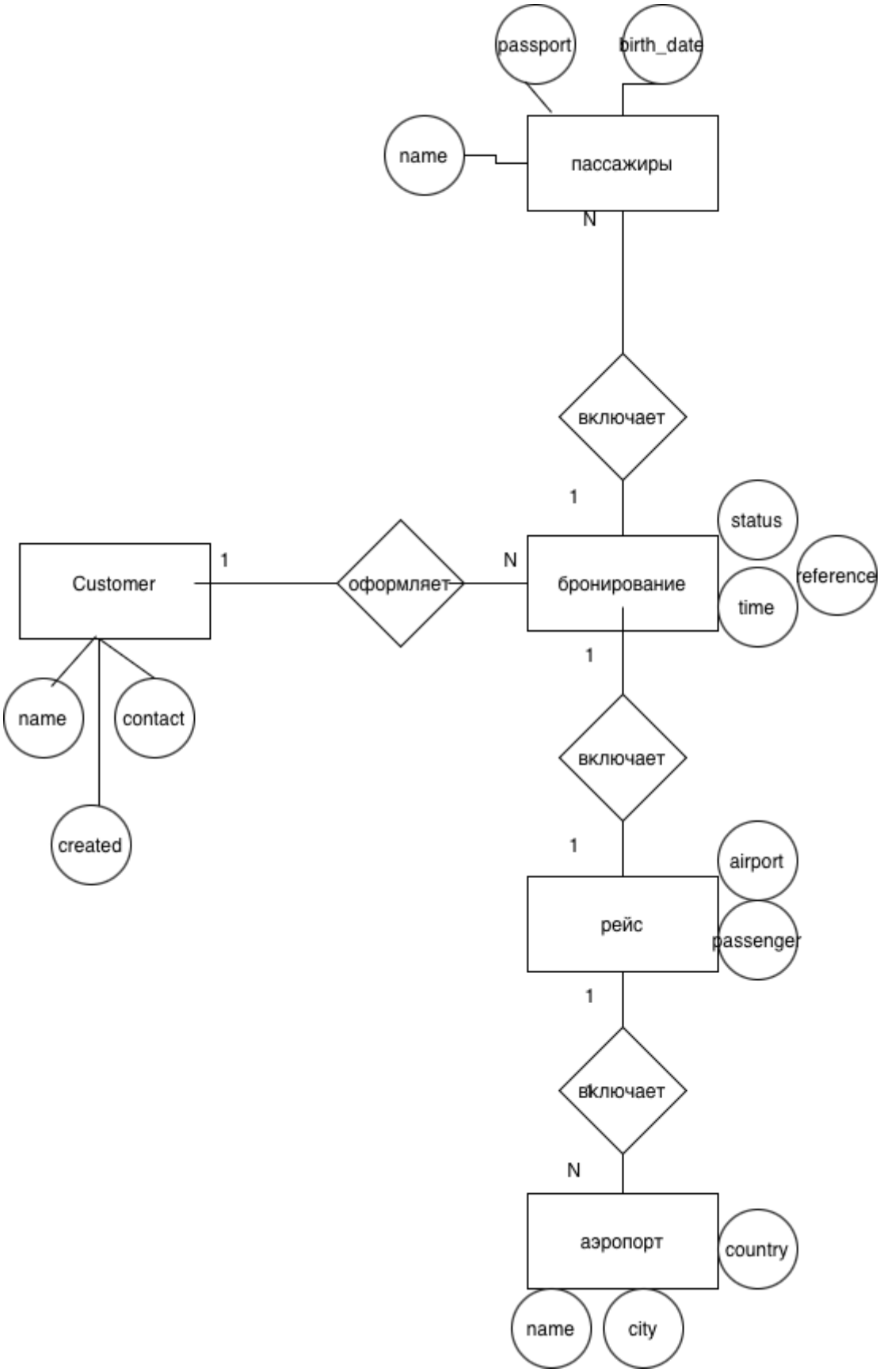
База данных

ER-диаграмма

Нотация Мартина


Customer
• id : integer [PK]
• first_name : varchar(100)
• last_name : varchar(100)
• middle_name : varchar(100)
• phone : varchar(20)
• email : varchar(30)
• created_at : timestamp
• updated_at : timestamp
оформляет 1:N
Booking
• booking_id : integer [PK]
• booking_reference : varchar(20)
• booking_time : timestamp
• status : varchar(50)
• customer_id : integer [FK]
• flight_id : integer [FK]
• created_at : timestamp
• updated_at : timestamp
включает
1:N
включает
1:1
Passenger
• passenger_id : integer [PK]
• first_name : varchar(100)
• last_name : varchar(100)
• middle_name : varchar(100)
• birth_date : date
• passport_number : varchar(50)
• booking_id : integer [FK]
Flight
• id : integer [PK]
• flight_number : varchar(20) [UK]
• airline_code : varchar(10)
• airline_name : varchar(255)
• airline_country : varchar(100)
• departure_time : timestamp
• arrival_time : timestamp
• duration_minutes : integer
• departure_airport_id : integer [FK]
• arrival_airport_id : integer [FK]
• created_at : timestamp
• updated_at : timestamp
вылетает/прилетает
1:N
Airport
• airport_id : integer [PK]
• name : varchar(255)
• city : varchar(100)
• country : varchar(100)
• created_at : timestamp
• updated_at : timestamp

Нотация Чена



Создание базы данных в PostgreSQL

База данных для системы бронирования авиабилетов

```
CREATE TABLE airports (
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    iata_code VARCHAR(10) NOT NULL UNIQUE,
    icao_code VARCHAR(10) NOT NULL UNIQUE,
    name VARCHAR(255) NOT NULL,
    city VARCHAR(100) NOT NULL,
    country VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


CREATE TABLE customers (
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    email VARCHAR(255) NOT NULL UNIQUE,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


CREATE TABLE flights (
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    flight_number VARCHAR(20) NOT NULL UNIQUE,
    airline_code VARCHAR(10) NOT NULL,
    airline_name VARCHAR(255) NOT NULL,
    airline_country VARCHAR(100) NOT NULL,
    departure_airport_id INTEGER NOT NULL,
    arrival_airport_id INTEGER NOT NULL,
    departure_time TIMESTAMP NOT NULL,
    arrival_time TIMESTAMP NOT NULL,
    duration_minutes INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_departure_airport
        FOREIGN KEY (departure_airport_id)
        REFERENCES airports(id)
        ON DELETE CASCADE,
    CONSTRAINT fk_arrival_airport
        FOREIGN KEY (arrival_airport_id)
        REFERENCES airports(id)
        ON DELETE CASCADE
);
```

```
CREATE TABLE bookings (
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    booking_reference VARCHAR(20) NOT NULL UNIQUE,
    customer_id INTEGER NOT NULL,
    flight_id INTEGER NOT NULL,
    booking_time TIMESTAMP NOT NULL,
    status VARCHAR(50) NOT NULL DEFAULT 'confirmed',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_booking_customer
        FOREIGN KEY (customer_id)
        REFERENCES customers(id)
        ON DELETE CASCADE,
    CONSTRAINT fk_booking_flight
        FOREIGN KEY (flight_id)
        REFERENCES flights(id)
        ON DELETE CASCADE
);

CREATE TABLE passengers (
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    booking_id INTEGER NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    birth_date DATE NOT NULL,
    passport_number VARCHAR(50) NOT NULL UNIQUE,
    nationality VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_passenger_booking
        FOREIGN KEY (booking_id)
        REFERENCES bookings(id)
        ON DELETE CASCADE
);
```

Вставка тестовых данных

```
-- Аэропорты
INSERT INTO airports (iata_code, icao_code, name, city, country) VALUES
('SVO', 'UUEE', 'Шереметьево', 'Москва', 'Россия'),
('DME', 'UUDD', 'Домодедово', 'Москва', 'Россия'),
('IST', 'LTFM', 'Стамбул', 'Стамбул', 'Турция'),
('FRA', 'EDDF', 'Франкфурт', 'Франкфурт', 'Германия'),
('CDG', 'LFPG', 'Шарль-де-Голль', 'Париж', 'Франция');

-- Клиенты
INSERT INTO customers (email, first_name, last_name, phone) VALUES
```

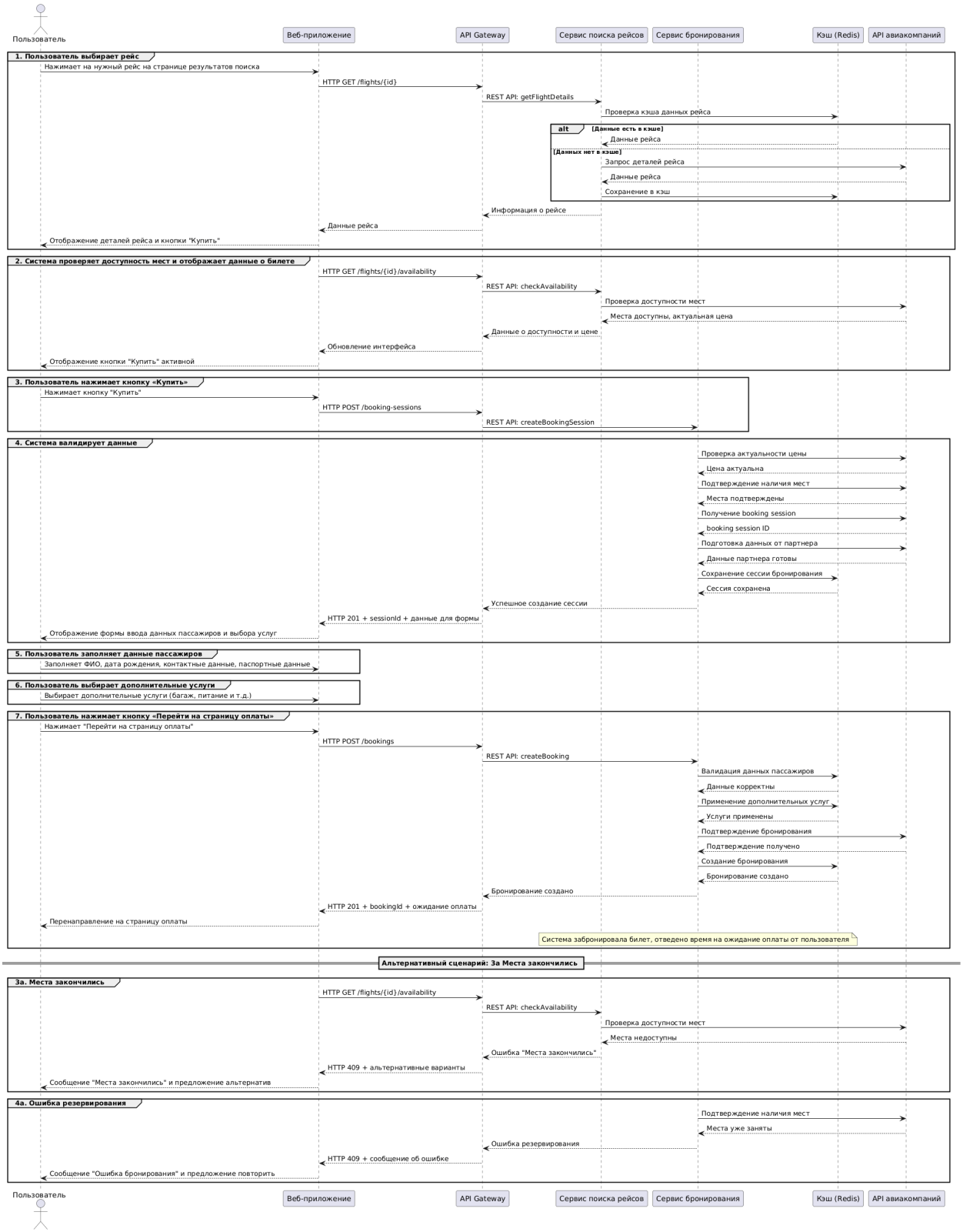
```
('ivanov@example.com', 'Иван', 'Иванов', '+79161234567'),
('petrova@example.com', 'Мария', 'Петрова', '+79162345678'),
('sidorov@example.com', 'Алексей', 'Сидоров', '+79163456789');

-- Рейсы (с информацией об авиакомпаниях)
INSERT INTO flights (flight_number, airline_code, airline_name, airline_country,
departure_airport_id, arrival_airport_id, departure_time, arrival_time, duration_minutes)
VALUES
('SU 1001', 'SU', 'Аэрофлот', 'Россия', 1, 3, '2024-03-15 08:00:00', '2024-03-15 11:30:00',
210),
('S7 2002', 'S7', 'S7 Airlines', 'Россия', 2, 4, '2024-03-16 14:30:00', '2024-03-16 16:45:00',
135),
('TK 3003', 'TK', 'Turkish Airlines', 'Турция', 3, 5, '2024-03-17 10:15:00', '2024-03-17
13:00:00', 165),
('LH 4004', 'LH', 'Lufthansa', 'Германия', 4, 1, '2024-03-18 19:00:00', '2024-03-18 23:30:00',
270);

-- Бронирования
INSERT INTO bookings (booking_reference, customer_id, flight_id, booking_time, status)
VALUES
('ABC123', 1, 1, '2024-03-10 15:30:00', 'confirmed'),
('DEF456', 2, 2, '2024-03-11 10:15:00', 'confirmed'),
('GHI789', 3, 3, '2024-03-12 12:45:00', 'pending');

-- Пассажиры
INSERT INTO passengers (booking_id, first_name, last_name, birth_date, passport_number,
nationality) VALUES
(1, 'Иван', 'Иванов', '1985-05-15', '1234567890', 'Россия'),
(1, 'Елена', 'Иванова', '1988-07-22', '0987654321', 'Россия'),
(2, 'Мария', 'Петрова', '1990-03-10', '1122334455', 'Россия'),
(3, 'Алексей', 'Сидоров', '1978-11-25', '5566778899', 'Россия'),
(3, 'Ольга', 'Сидорова', '1980-09-18', '6677889900', 'Россия'),
(3, 'Дмитрий', 'Сидоров', '2010-12-05', '7788990011', 'Россия');
```

Пример запроса

```
SELECT
    b.booking_reference,
    c.first_name || ' ' || c.last_name AS customer_name,
    f.flight_number,
    f.airline_name AS airline,
    dep.iata_code AS departure_airport,
    arr.iata_code AS arrival_airport,
    f.departure_time,
    f.arrival_time,
    b.booking_time,
    b.status,
    COALESCE(p.passengers_count, 0) AS passengers_count
FROM bookings b
JOIN customers c ON b.customer_id = c.id
JOIN flights f ON b.flight_id = f.id
JOIN airports dep ON f.departure_airport_id = dep.id
JOIN airports arr ON f.arrival_airport_id = arr.id
```

```
LEFT JOIN (
    SELECT booking_id, COUNT(*) AS passengers_count
    FROM passengers
    GROUP BY booking_id
) p ON p.booking_id = b.id
ORDER BY b.booking_time DESC;
```

Диаграмма последовательности



Пользовательский сценарий «Бронирование билета»

Описание эндпойнтов

Метод	Эндпойнт	Описание
GET	/flights/{id}	Получение информации о рейсе
POST	/booking-sessions	Создание сессии бронирования с проверкой доступности мест и резервированием на 15 минут
POST	/bookings	Создание окончательного бронирования с данными пассажиров, дополнительными услугами и контактной информацией

Последовательность действий (Flow):

1. Пользователь выбирает рейс FL-12345
2. Получение сведений о рейсе GET /flights/{id}
2. Создание сессии POST /booking-sessions – получаем sessionId
3. Заполнение данных POST /bookings с sessionId и данными пассажиров – получаем bookingId и paymentUrl

Название метода:

● POST /booking-sessions
Content-Type: application/json
Authorization: Bearer <token>

Что делает метод: создает сессию бронирования

Входные параметры (пример запроса):

- Body:

```
{
 "flightId": "FL-12345",
 "departureDate": "2024-03-15",
 "passengersCount": 2,
 "cabinClass": "ECONOMY"
}
```

Выходные параметры:

- успех:

○ Код ответа: 201

○ Пример:


```
{
 "sessionId": "sess_abc123def456",
 "expiresAt": "2024-03-15T10:45:00Z",
 "flightDetails": {
  "flightId": "FL-12345",
  "airline": "Аэрофлот",
  "departure": "2024-03-15T08:00:00Z",
  "arrival": "2024-03-15T11:30:00Z",
  "from": "SVO",
  "to": "LED",
  "cabinClass": "ECONOMY"
 },
 "price": {
  "total": 15000.00,
  "currency": "RUB",
  "breakdown": {
   "baseFare": 12000.00,
   "taxes": 3000.00
  }
 }
}
```


Ошибка: «Места закончились»
○ Код ответа: 409

○ Пример:


```
{
 "code": "NO_SEATS_AVAILABLE",
 "message": "Места на выбранном рейсе закончились",
 "details": {
  "flightId": "FL-12345",
  "alternativeFlights": [
   {
    "flightId": "FL-12346",
    "departure": "2024-03-15T10:00:00Z",
    "price": 16000.00
   }
  ]
 }
}
```


Ошибка: некорректные параметры (400 Bad Request)


```
{
 "code": "VALIDATION_ERROR",
 "message": "Некорректные параметры запроса",
```

```
  "details": {
    "field": "passengersCount",
    "issue": "Максимум 9 пассажиров в одном бронировании"
  }
}
```

2. Создание бронирования

POST /bookings
Content-Type: application/json
Authorization: Bearer <token>

Тело запроса:

```
{
  "sessionId": "sess_abc123def456",
  "passengers": [
    {
      "firstName": "Иван",
      "lastName": "Иванов",
      "middleName": "Иванович",
      "birthDate": "1985-05-15",
      "documentType": "PASSPORT",
      "documentNumber": "123456789",
      "nationality": "RU",
      "contactEmail": "ivan@example.com",
      "contactPhone": "+79161234567"
    },
    {
      "firstName": "Мария",
      "lastName": "Иванова",
      "middleName": "Петровна",
      "birthDate": "1988-07-22",
      "documentType": "PASSPORT",
      "documentNumber": "987654321",
      "nationality": "RU",
      "contactEmail": "maria@example.com",
      "contactPhone": "+79167654321"
    }
  ],
  "additionalServices": [
    {
      "serviceId": "baggage_20kg",
      "quantity": 2
    },
    {
      "serviceId": "priority_boarding",
      "quantity": 1
    }
  ],
  "contactInfo": {
    "email": "ivan@example.com",
    "phone": "+79161234567"
  },
```

```
  "specialRequests": "Выбор места"
}
```


Успешный ответ (201 Created)


```
{
  "bookingId": "book_xyz789abc456",
  "status": "PENDING_PAYMENT",
  "expiresAt": "2024-03-15T11:00:00Z",
  "passengers": [
    {
      "passengerId": "pass_1",
      "firstName": "Иван",
      "lastName": "Иванов",
      "middleName": "Иванович",
      "birthDate": "1985-05-15",
      "documentType": "PASSPORT",
      "documentNumber": "123456789",
      "nationality": "RU",
      "contactEmail": "ivan@example.com",
      "contactPhone": "+79161234567",
      "seat": "15A"
    },
    {
      "passengerId": "pass_2",
      "firstName": "Мария",
      "lastName": "Иванова",
      "middleName": "Петровна",
      "birthDate": "1988-07-22",
      "documentType": "PASSPORT",
      "documentNumber": "987654321",
      "nationality": "RU",
      "contactEmail": "maria@example.com",
      "contactPhone": "+79167654321",
      "seat": "15B"
    }
  ],
  "flightDetails": {
    "flightId": "FL-12345",
    "airline": "Аэрофлот",
    "departure": "2024-03-15T08:00:00Z",
    "arrival": "2024-03-15T11:30:00Z",
    "from": "SVO",
    "to": "LED",
    "cabinClass": "ECONOMY"
  },
  "price": {
    "total": 18500.00,
    "currency": "RUB",
    "breakdown": {
      "baseFare": 12000.00,
      "taxes": 3000.00,
      "additionalServices": 3500.00
```

```
    }
  },
  "paymentUrl": "https://payment.flightbooking.com/pay/book_xyz789abc456"
}
```

Ошибка валидации данных (400 Bad Request)

```
{
  "code": "PASSENGER_DATA_INVALID",
  "message": "Данные пассажиров некорректны",
  "details": [
    {
      "field": "passengers[0].documentNumber",
      "issue": "Неверный формат паспортных данных"
    },
    {
      "field": "passengers[1].birthDate",
      "issue": "Дата рождения должна быть в прошлом"
    }
  ]
}
```

Ошибка: сессия истекла (404 Not Found)

```
{
  "code": "SESSION_EXPIRED",
  "message": "Сессия бронирования истекла или не найдена",
  "details": {
    "sessionId": "sess_abc123def456"
  }
}
```

OpenAPI

```yaml
openapi: 3.0.3
info:
  title: Flight Booking API
  description: |
    REST API для бронирования авиабилетов.
    Соответствует структуре базы данных с 5 сущностями.
  version: '1.0.0'
  contact:
    name: Flight Booking Support
    email: support@flightbooking.com
  license:
    name: MIT
    url: https://opensource.org/licenses/MIT

servers:
  - url: https://api.flightbooking.com/v1
```

```yaml
    description: Production server

paths:
  /booking-sessions:
    post:
      summary: Создание сессии бронирования
      description: |
        Создает сессию бронирования для выбранного рейса.
        Резервирует места на 15 минут.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/BookingSessionRequest'
      responses:
        '201':
          description: Сессия бронирования создана
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/BookingSessionResponse'
        '400':
          description: Некорректный запрос
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '404':
          description: Рейс не найден
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '409':
          description: Конфликт (места закончились)
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '500':
          description: Внутренняя ошибка сервера
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'

  /bookings:
    post:
      summary: Создание бронирования
      description: |
        Создает окончательное бронирование на основе сессии.
        Валидирует данные пассажиров и создает запись в системе.
```

```yaml
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/BookingRequest'
    responses:
      '201':
        description: Бронирование создано
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/BookingResponse'
      '400':
        description: Ошибка валидации данных
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ValidationError'
      '404':
        description: Сессия не найдена или истекла
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '409':
        description: Ошибка резервирования
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '422':
        description: Необрабатываемая сущность
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Внутренняя ошибка сервера
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'

components:
  schemas:
    BookingSessionRequest:
      type: object
      required:
        - flightId
        - departureDate
        - passengersCount
      properties:
```

```yaml
      flightId:
        type: integer
        description: Идентификатор рейса (соответствует flights.id)
      departureDate:
        type: string
        format: date
        description: Дата вылета
      passengersCount:
        type: integer
        minimum: 1
        maximum: 9
        description: Количество пассажиров
      cabinClass:
        type: string
        description: Класс обслуживания
        enum: [ECONOMY, PREMIUM_ECONOMY, BUSINESS, FIRST]
        default: ECONOMY

  BookingSessionResponse:
    type: object
    required:
      - sessionId
      - expiresAt
      - flightDetails
      - price
    properties:
      sessionId:
        type: string
        description: Идентификатор сессии бронирования
      expiresAt:
        type: string
        format: date-time
        description: Время истечения сессии
      flightDetails:
        $ref: '#/components/schemas/FlightDetails'
      price:
        $ref: '#/components/schemas/PriceDetails'

  BookingRequest:
    type: object
    required:
      - sessionId
      - passengers
      - contactInfo
    properties:
      sessionId:
        type: string
        description: Идентификатор сессии бронирования
      passengers:
        type: array
        minItems: 1
        maxItems: 9
        items:
```

```yaml
          $ref: '#/components/schemas/Passenger'
        description: Данные пассажиров
      contactInfo:
        $ref: '#/components/schemas/ContactInfo'
      specialRequests:
        type: string
        maxLength: 500
        description: Особые пожелания

  BookingResponse:
    type: object
    required:
      - bookingId
      - status
      - expiresAt
      - passengers
      - flightDetails
      - price
    properties:
      bookingId:
        type: integer
        description: Идентификатор бронирования (соответствует bookings.id)
      status:
        type: string
        description: Статус бронирования
        enum: [PENDING_PAYMENT, CONFIRMED, CANCELLED, EXPIRED]
      expiresAt:
        type: string
        format: date-time
        description: Время истечения бронирования для оплаты
      passengers:
        type: array
        items:
          $ref: '#/components/schemas/PassengerWithId'
        description: Пассажиры
      flightDetails:
        $ref: '#/components/schemas/FlightDetails'
      price:
        $ref: '#/components/schemas/PriceDetails'
      paymentUrl:
        type: string
        format: uri
        description: URL для оплаты бронирования

  Passenger:
    type: object
    required:
      - firstName
      - lastName
      - birthDate
      - documentType
      - documentNumber
      - nationality
```

```yaml
      properties:
        firstName:
          type: string
          description: Имя
        lastName:
          type: string
          description: Фамилия
        middleName:
          type: string
          description: Отчество
        birthDate:
          type: string
          format: date
          description: Дата рождения
        documentType:
          type: string
          description: Тип документа
          enum: [PASSPORT, ID_CARD, BIRTH_CERTIFICATE, OTHER]
        documentNumber:
          type: string
          description: Номер документа
        nationality:
          type: string
          description: Гражданство (код страны ISO 3166-1 alpha-2)
        contactEmail:
          type: string
          format: email
          description: Контактный email
        contactPhone:
          type: string
          description: Контактный телефон

    PassengerWithId:
      allOf:
        - $ref: '#/components/schemas/Passenger'
        - type: object
          properties:
            passengerId:
              type: integer
              description: Идентификатор пассажира в системе (соответствует
passengers.id)

    ContactInfo:
      type: object
      required:
        - email
        - phone
      properties:
        email:
          type: string
          format: email
          description: Контактный email (соответствует customers.email)
        phone:
```

```yaml
        type: string
        description: Контактный телефон (соответствует customers.phone)

    FlightDetails:
      type: object
      required:
        - flightId
        - flightNumber
        - airlineCode
        - airlineName
        - departureAirportCode
        - arrivalAirportCode
        - departureTime
        - arrivalTime
      properties:
        flightId:
          type: integer
          description: Идентификатор рейса (соответствует flights.id)
        flightNumber:
          type: string
          description: Номер рейса (соответствует flights.flight_number)
        airlineCode:
          type: string
          description: Код авиакомпании (соответствует flights.airline_code)
        airlineName:
          type: string
          description: Название авиакомпании (соответствует flights.airline_name)
        departureAirportCode:
          type: string
          description: Код аэропорта вылета IATA (соответствует airports.iata_code)
        arrivalAirportCode:
          type: string
          description: Код аэропорта прилета IATA (соответствует airports.iata_code)
        departureTime:
          type: string
          format: date-time
          description: Время вылета (соответствует flights.departure_time)
        arrivalTime:
          type: string
          format: date-time
          description: Время прилета (соответствует flights.arrival_time)
        cabinClass:
          type: string
          description: Класс обслуживания
          enum: [ECONOMY, PREMIUM_ECONOMY, BUSINESS, FIRST]

    PriceDetails:
      type: object
      required:
        - total
        - currency
      properties:
        total:
```

```yaml
        type: number
        format: float
        description: Общая стоимость
      currency:
        type: string
        description: Валюта (код ISO 4217)
      breakdown:
        type: object
        description: Детализация стоимости
        properties:
          baseFare:
            type: number
            description: Базовая стоимость
          taxes:
            type: number
            description: Налоги и сборы

  Error:
    type: object
    required:
      - code
      - message
    properties:
      code:
        type: string
        description: Код ошибки
      message:
        type: string
        description: Сообщение об ошибке
      details:
        type: object
        description: Детали ошибки
        additionalProperties: true

  ValidationError:
    type: object
    required:
      - code
      - message
      - details
    properties:
      code:
        type: string
        description: Код ошибки валидации
      message:
        type: string
        description: Сообщение об ошибке
      details:
        type: array
        description: Список ошибок валидации
        items:
          type: object
          properties:
```

```yaml
            field:
              type: string
              description: Поле с ошибкой
            issue:
              type: string
              description: Описание проблемы
```