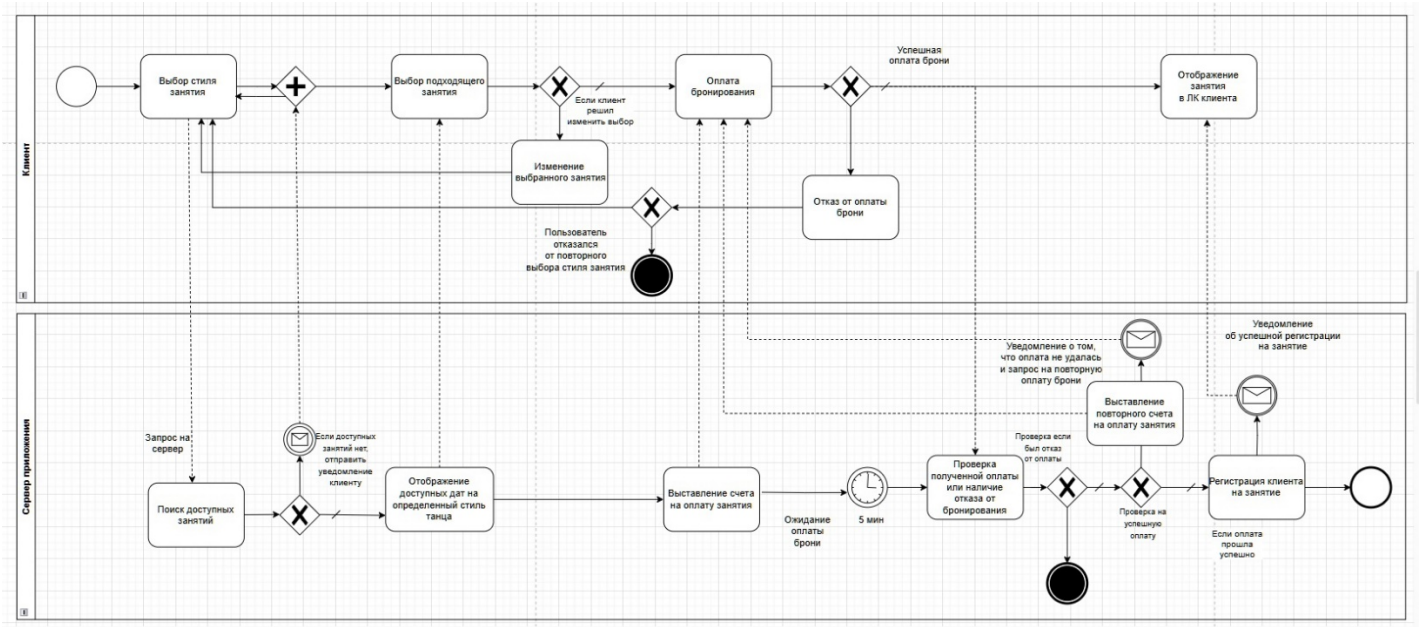
Сироткина Маргарита Сергеевна «Аналитика»

Тема задания: Описание мобильного клиент-серверного приложения «The Dance Lab» (приложения для бронирования занятий в танцевальной студии).

1. Бизнес-процесс создания заказа (Draw.io)

В качестве задания была выбрана танцевальная студия.
Описание работы приложения по бронированию занятий в танцевальной студии представлено в бизнес-модели:



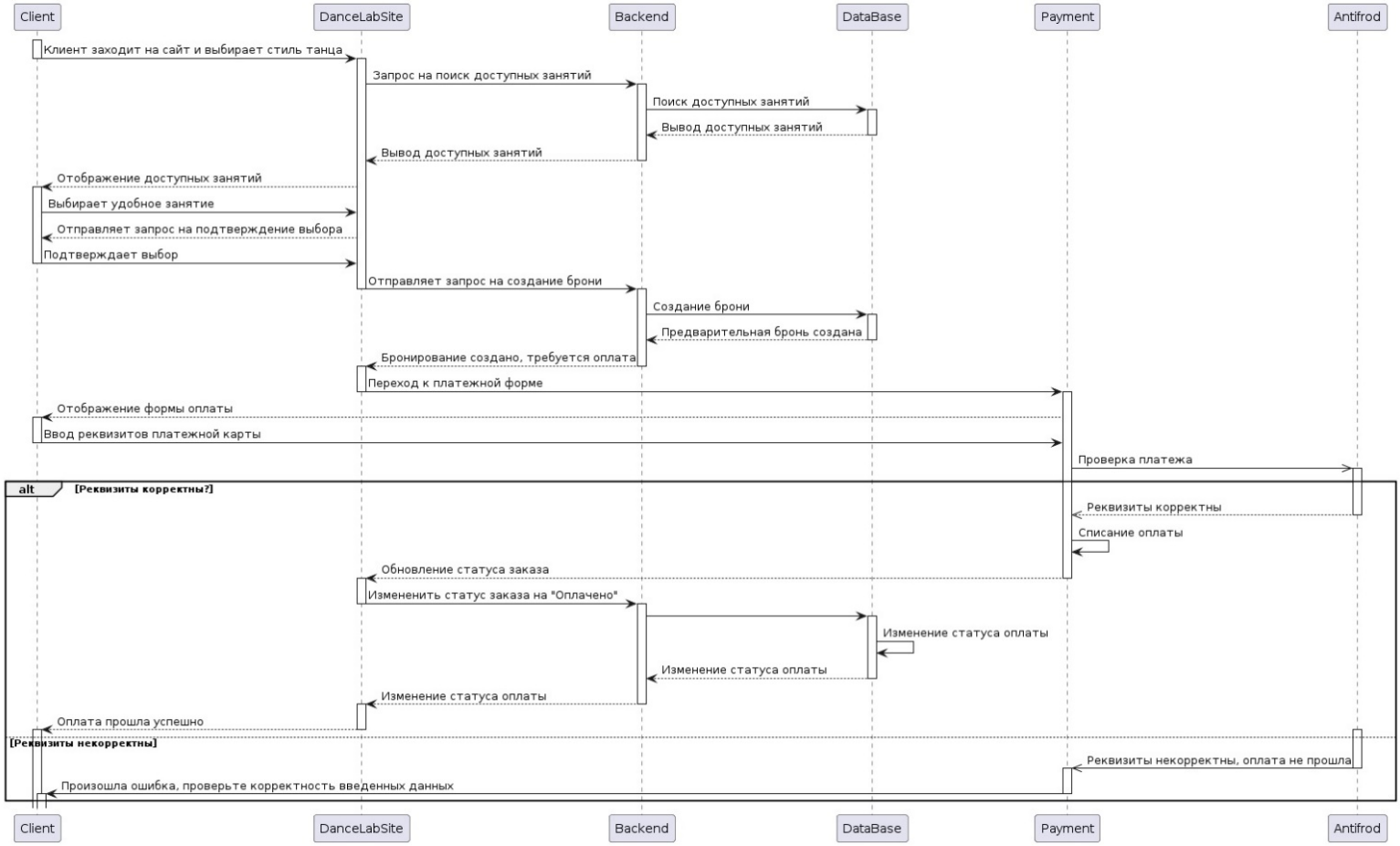
«The Dance Lab» - это мобильное приложение, разработанное для удобного бронирования занятий в танцевальной студии. Приложение предназначено для клиентов, желающих записаться на занятия различных стилей танцев.

Краткое описание бизнес-модели: Клиент выбирает интересующий стиль танцев, тогда система отображает доступные позиции. После выбора интересующей позиции, пользователь оплачивает забронированное место на занятии, оно отображается в его личном кабинете с определенным статусом (в данном случае «Оплачено»).

Система проверяет какие есть доступные занятия по выбранному стилю, предлагает их пользователю и для выбранных позиций предоставляет счет на оплату. Далее проверяет был ли оплачен заказ и не было ли отказа клиента от оплаты.

2. Описание процесса синхронизации данных между клиентом и сервером (создание, редактирование и отмена заказа, изменение персональных данных, оплата заказа и т.д.). Представить все в диаграммах UML, API методах и других представлениях.

Создание и оплата заказа (UML):



Клиент:

Пользователь выбирает стиль и понравившееся занятие.

Приложение формирует запрос на сервер, передавая данные о выбранном занятии (идентификатор занятия, идентификатор пользователя).

Данные могут также включать дополнительные параметры, такие как дата и время бронирования.

Сервер:

Сервер получает запрос от клиента и проверяет его на валидность (например, наличие свободных мест на занятии).

Если данные верны, сервер создает новую запись о бронировании в базе данных, связывая пользователя с выбранным занятием.

Сервер возвращает клиенту подтверждение успешного бронирования и выставляет счет на оплату бронирования, в случае успеха, подтверждает бронь и изменяет статус бронирования в личном кабинете пользователя.

Некоторые API методы, используемые в приложении:

1. Создание заказа (бронирование занятия)

HTTP метод: POST

Пример запроса (JSON):

```
{
  "customer_id": 123,
  "class_id": 456,
  "booking_date": "2024-07-10T10:00:00"
}
```

Пример ответа (JSON) при успешном создании:

Сервер возвращает уникальный идентификатор бронирования и статус операции.

```
{
  "booking_id": 789,
  "status": "success",
  "message": "Booking created successfully."
}
```

2. Оплата заказа

HTTP метод: POST

Пример запроса (JSON):

```
{
  "booking_id": 789,
  "amount": 500.00,
  "payment_method": "credit_card",
  "card_details": {
    "card_number": "1234567812345678",
    "expiry_date": "07/26",
    "cvv": "123"
  }
```

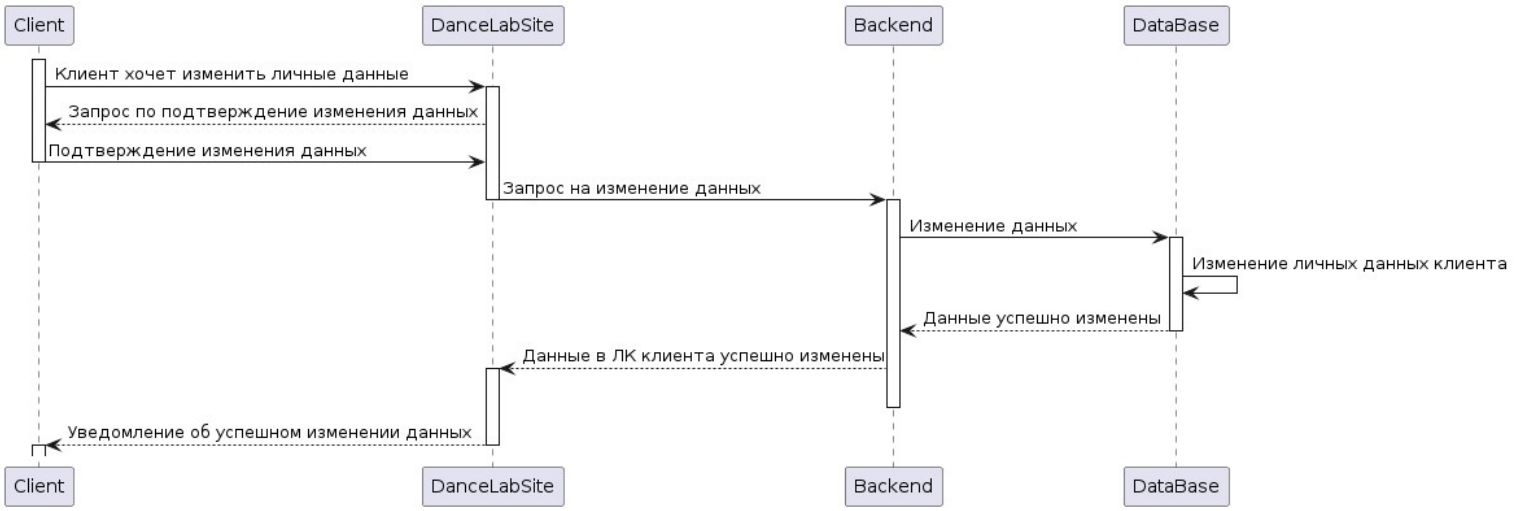
}

Пример ответа (JSON) при успешной оплате:


```
{
  "status": "success",
  "message": "Payment processed successfully."
}
```


Этот метод инициирует процесс оплаты для указанного бронирования на указанную сумму. Сервер возвращает статус операции и сообщение о результате.

Редактирование личной информации в профиле (UML):



3. Редактирование профиля пользователя

HTTP метод: PATCH

Пример запроса (JSON):

```
{
  "customer_id": 123,
  "first_name": "Иван",
  "last_name": "Иванов",
  "phone_number": "+7837567890",
}
```

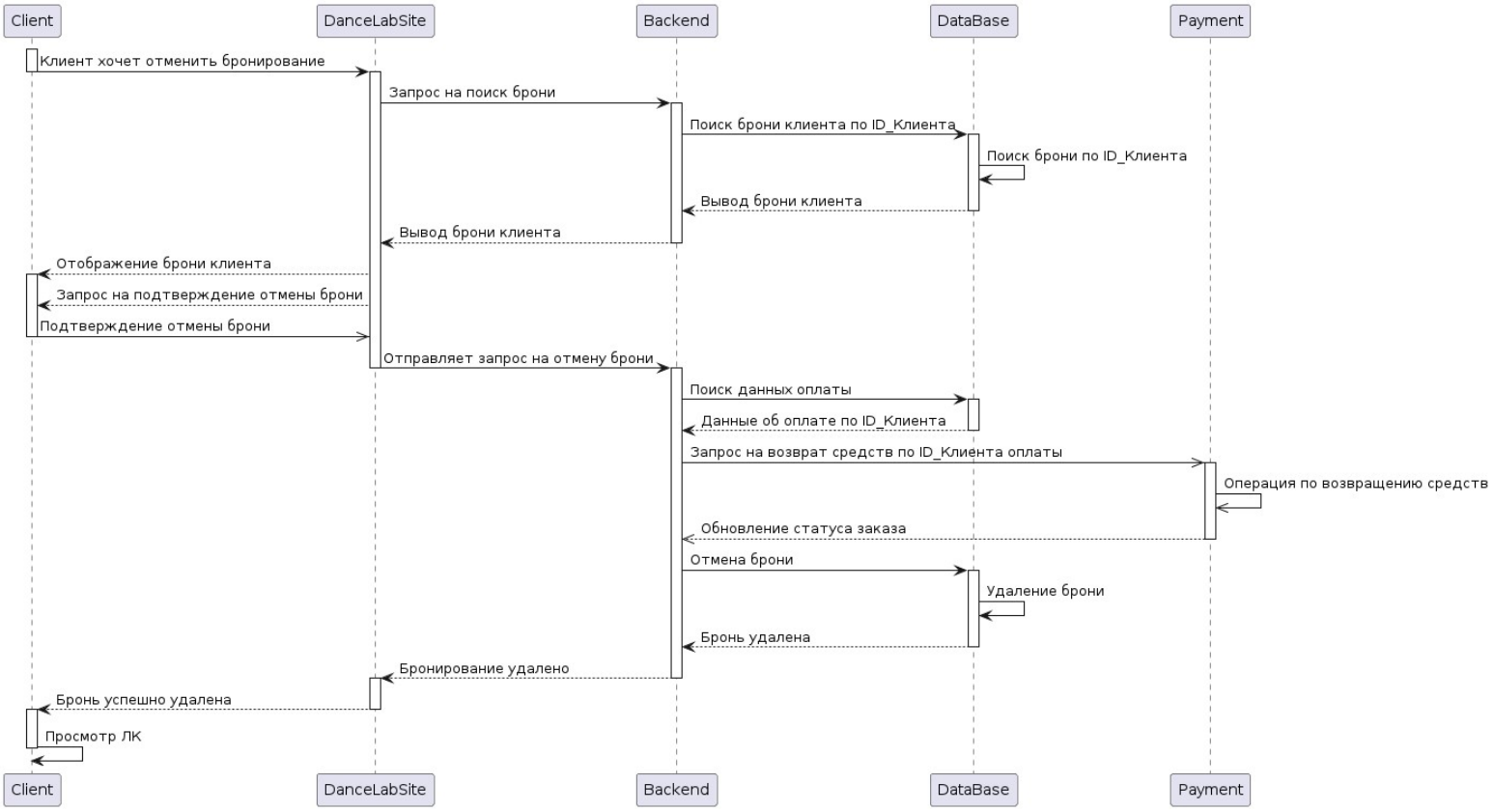
Пример ответа (JSON) при успешном выполнении:

```
{
  "status": "success",
  "message": "Profile updated successfully."
}
```

Этот метод позволяет клиенту обновлять персональные данные в своем профиле.

Сервер обновляет соответствующую запись в базе данных и возвращает подтверждение успешного обновления.

Отмена бронирования (UML):



4. Отмена бронирования

HTTP метод: PUT

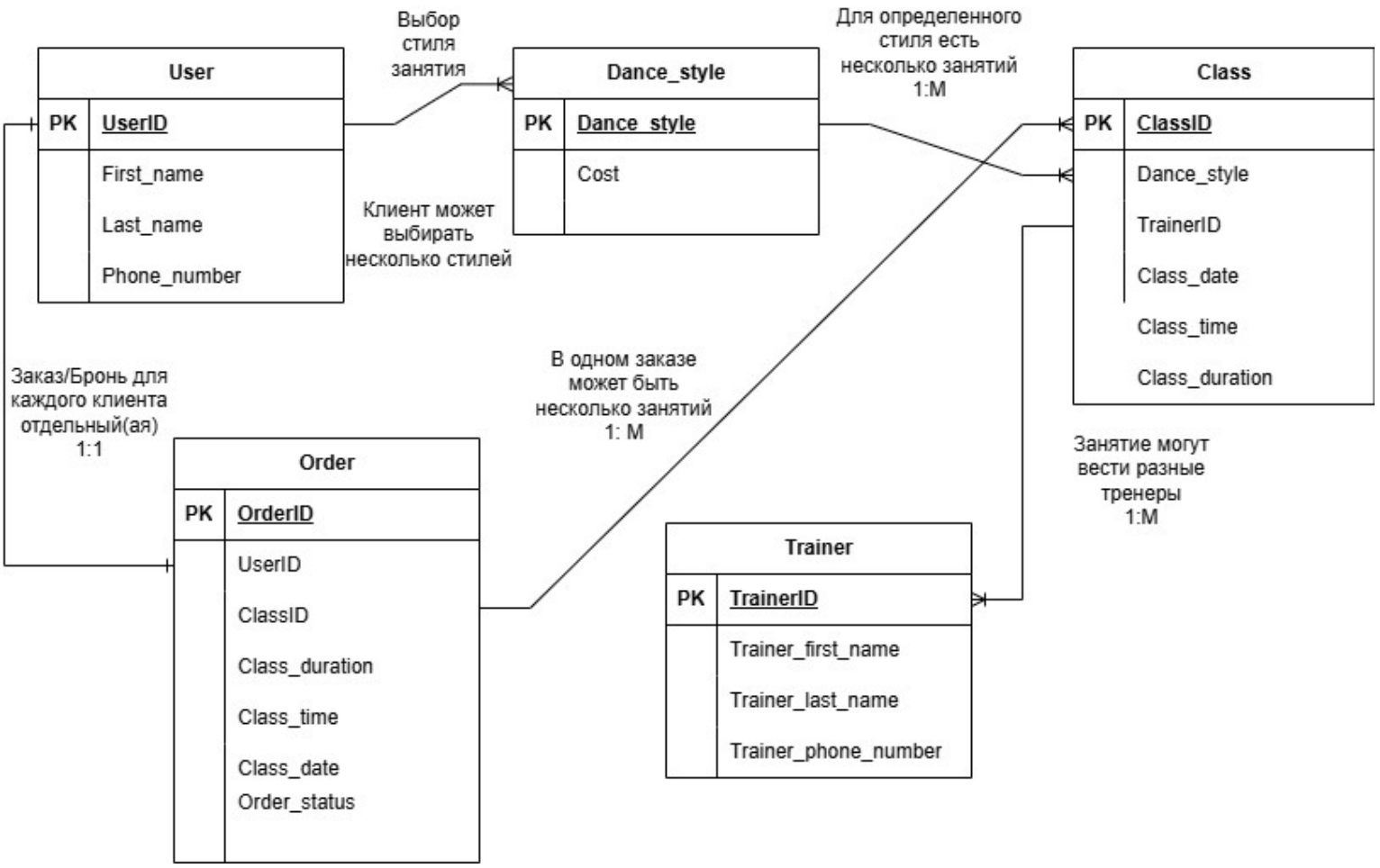
Пример запроса (JSON) для отмены занятия:


```
{
  "booking_id": 789,
  "status": "cancelled"
}
```


Пример ответа (JSON) при успешном изменении:


```
{
  "status": "success",
  "message": "Booking updated successfully."
}
```

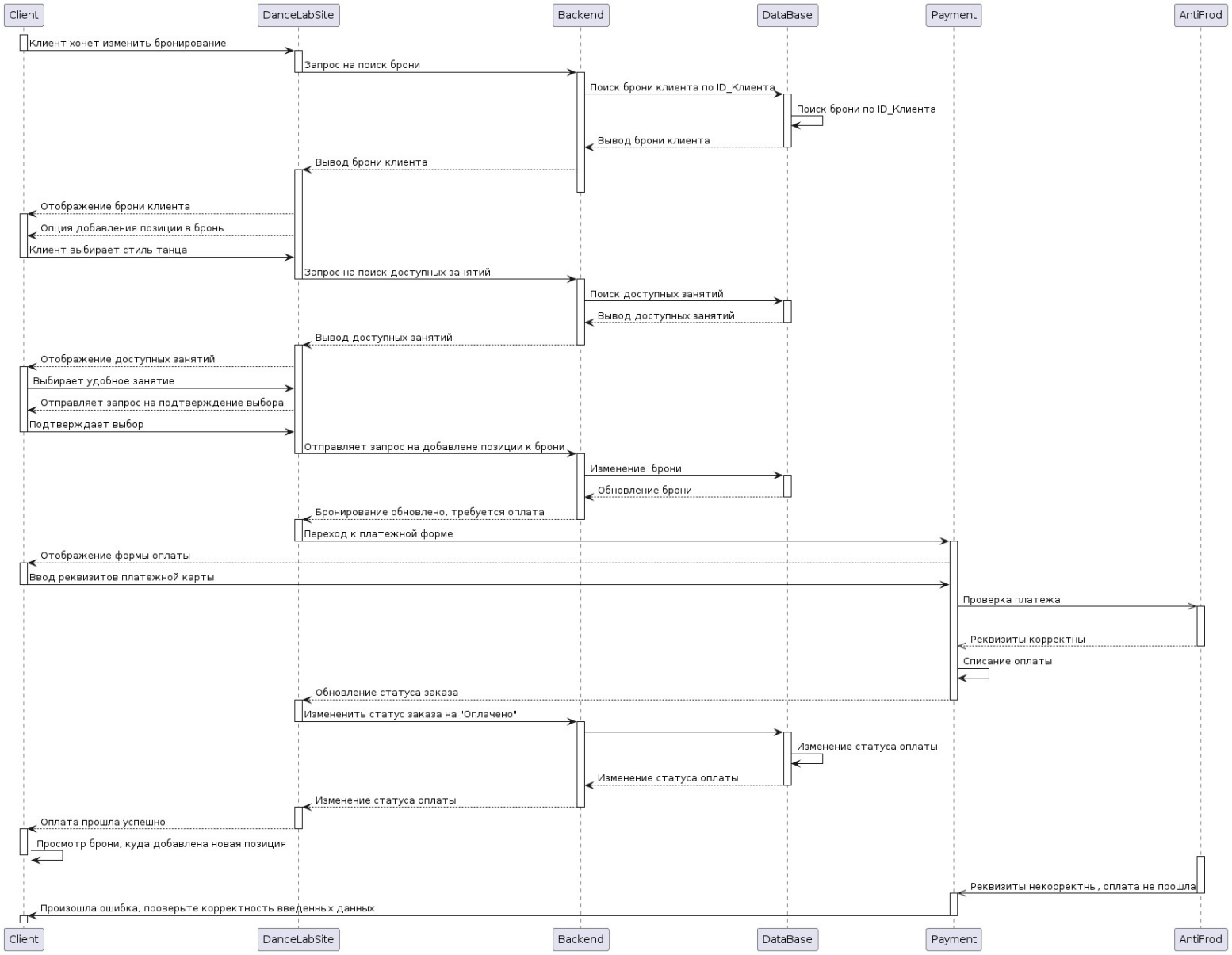
ER-диаграмма сущностей



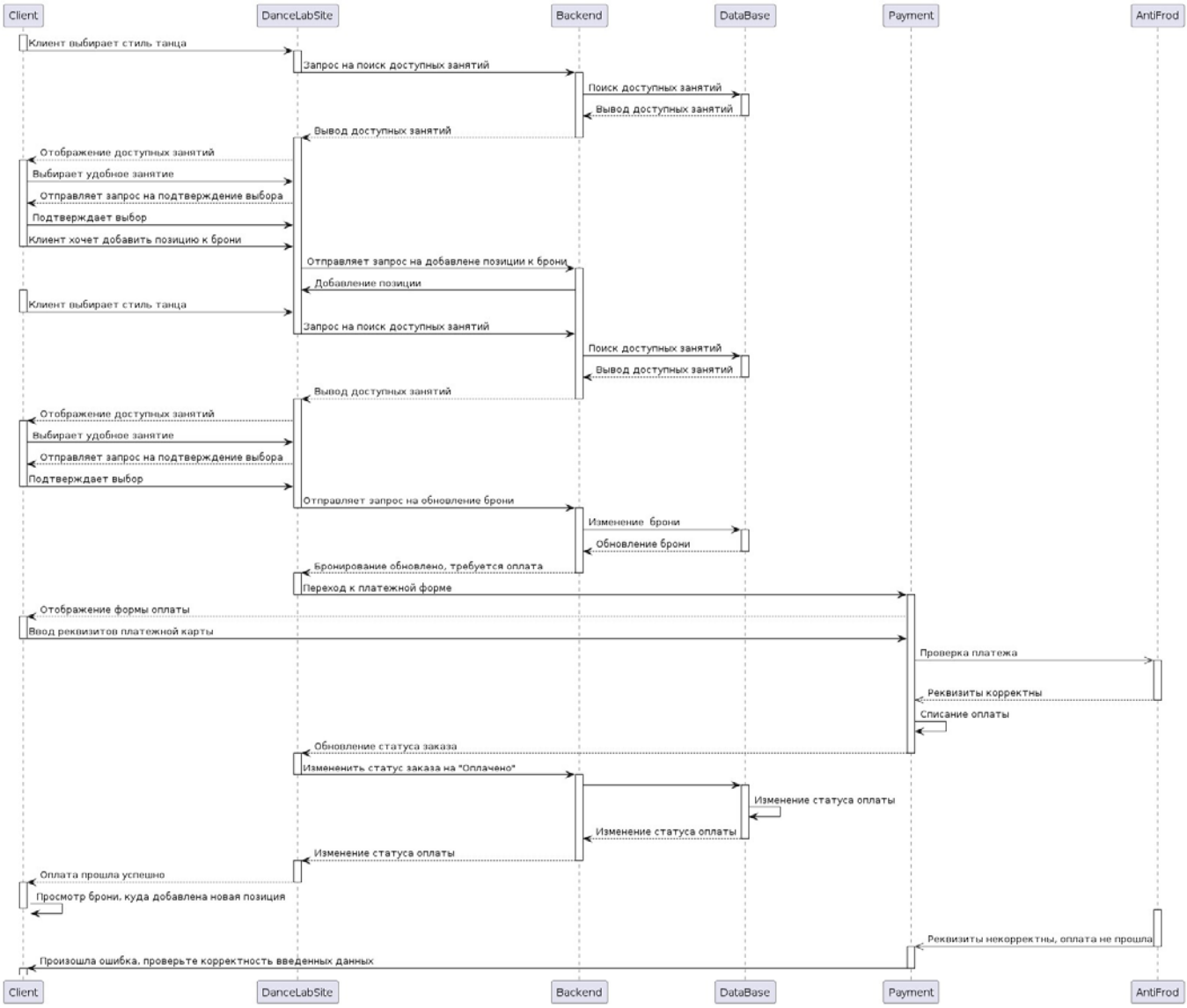
Хранение информации:

- User информация о пользователе (ID_пользователя, Имя, Фамилия, Номер телефона);
- Dance_style информация о стиле танцевального занятия (Танцевальный стиль, Стоимость);
- Class информация о танцевальном занятии (ID_занятия, ID_тренера, Дата проведения занятия, Время проведения занятия, Длительность занятия);
- Trainer информация о тренере (ID_тренера, Имя тренера, Фамилия тренера, Номер телефона тренера);
- Order информация о бронировании (ID_брони, ID_пользователя, ID_занятия, Длительность занятия, Время проведения занятия, Дата проведения занятия, Статус брони).

4. Редактирование заказа (UML):



Редактирование заказа – добавление позиции к уже существующей брони, оплата только за добавленную позицию, так как предыдущая бронь уже была оплачена. (Проверка оплаты через рамку alt, как в первом примере не отображается, но в коде она есть)



Редактирование заказа, добавление позиции к еще не оплаченной брони (Проверка оплаты через рамку alt, как в первом примере не отображается, но в коде она есть).

Выше приведены примеры редактирования заказа, по аналогии строятся UML модели редактирования заказа в случае удаления позиции из списка:

Если оплата еще не была совершена, то запрос, подтвержденный пользователем, об удалении выбранной позиции, отправляется на сервер, где обрабатывается и удаляется из предварительного бронирования.

Если оплата уже была совершена, то сервер находит бронь и платежные данные пользователя в системе, и, после подтверждения, отправляет запрос в

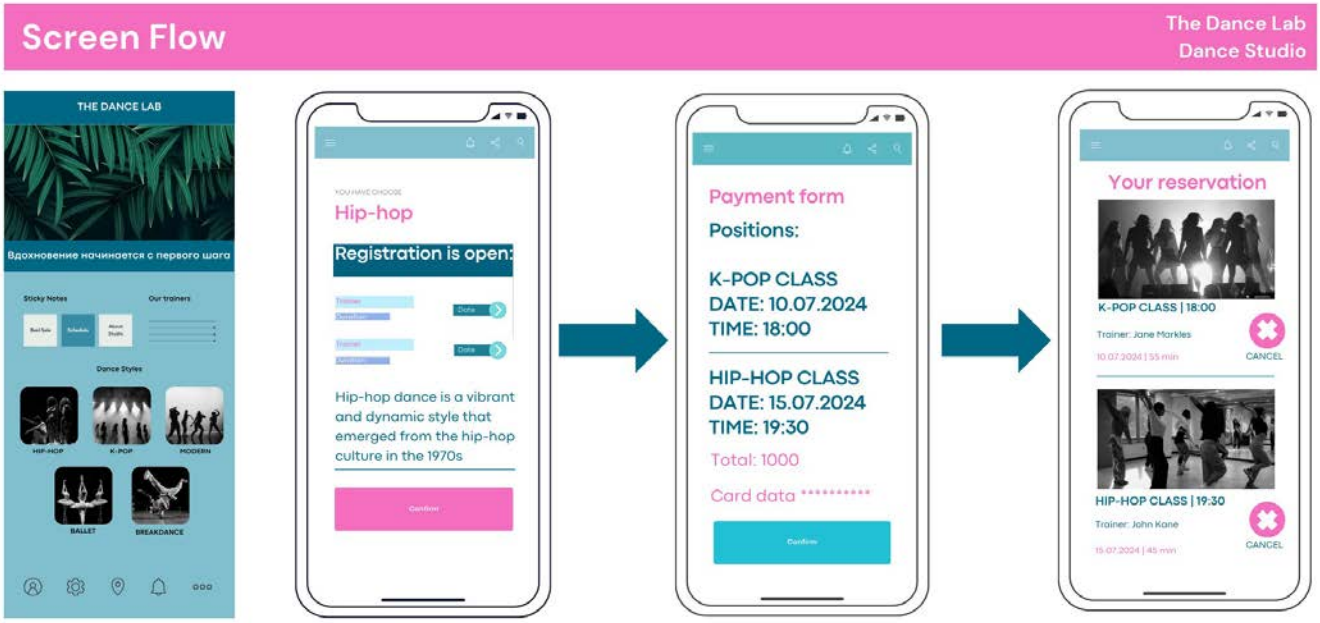
платежную систему для возврата средств за выбранную позицию и удаляет ее из заказа в базе данных.

Процесс редактирования заказа:

1) Пользователь отправляет запрос на изменение заказа.
2) Добавляет или убирает позиции в заказе.
3) Клиентское приложение отправляет запрос на сервер с обновленными данными заказа.
4) Сервер обрабатывает запрос, обновляет информацию в базе данных и возвращает подтверждение.
5) Клиентское приложение получает подтверждение и отображает обновленную информацию заказа пользователю.

***(Получилось не совсем реалистичное приложение, так как можно вернуть часть стоимости при отмене позиции, но очень полезное для пользователей. В рамках практики мне было интересно попробовать реализовать такой формат).

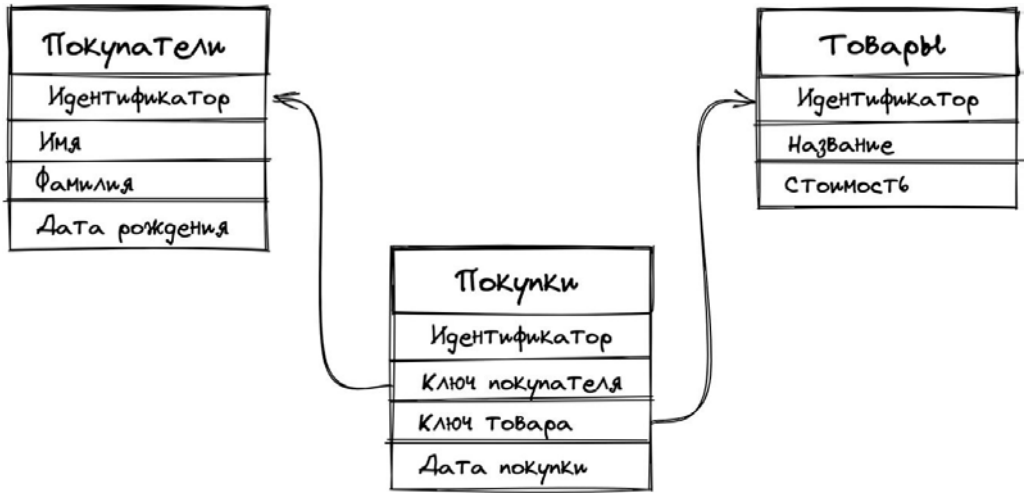
3. Прототип одного из экранов данного мобильного приложения и описать пользовательский интерфейс для данного.



На данном прототипе изображен главный экран мобильного приложения «The Dance Lab», где пользователь выбирает интересующие его танцевальные направления. После выбора определенного стиля пользователь переходит в раздел с доступными занятиями по данному направлению. В полях указаны тренер, длительность и даты, которые можно выбрать, нажав на стрелочки справа. Также краткая информация о стиле танца и кнопка, подтверждающая выбор пользователя.
Далее клиент переходит к заполнению платежной формы, где видит выбранные позиции, итоговую сумму и заполняет данные кредитной карты, либо, если он уже совершал покупки по этой карте в приложении, то она автоматически заполняет поле и предлагает оплатить с той же карты, которая есть в системе.

После успешной оплаты клиент получает сообщение о завершении операции и смене статуса заказа на «Оплачено». В его личном кабинете отражаются забронированные позиции, которые он может в любой момент отменить, нажав на крестик справа напротив выбранной позиции.

5. Написать SQL-запросы для данной реляционной модели:



1) Вывести покупателей с количеством осуществленных покупок:

SELECT Покупатели.Идентификатор Покупатели.Имя, Покупатели.Фамилия, COUNT(Покупки.Идентификатор)

FROM Покупатели JOIN Покупки ON

Покупатели.Идентификатор=Покупки.Ключ_покупателя

GROUP BY Покупатели.Идентификатор

2) Общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания:

SELECT Покупатели.Идентификатор, Покупатели.Имя, Покупатели.Фамилия, SUM(Товары.Стоимость)

FROM Покупатели JOIN Покупки ON Покупатели.Идентификатор=Покупки.Ключ_покупателя

JOIN Товары ON Товары.Идентификатор=Покупки.Ключ_товара

GROUP BY Покупатели.Идентификатор

ORDER BY DESC

3) Получить покупателей, купивших только один товар:

SELECT Покупатели.Идентификатор Покупатели.Имя, Покупатели.Фамилия

Покупатели JOIN Покупки ON Покупатели.Идентификатор=Покупки.Ключ_покупателя

JOIN Товары ON Покупки.Ключ_товара=Товары.Идентификатор

HAVING COUNT(Покупки.Идентификатор)=1