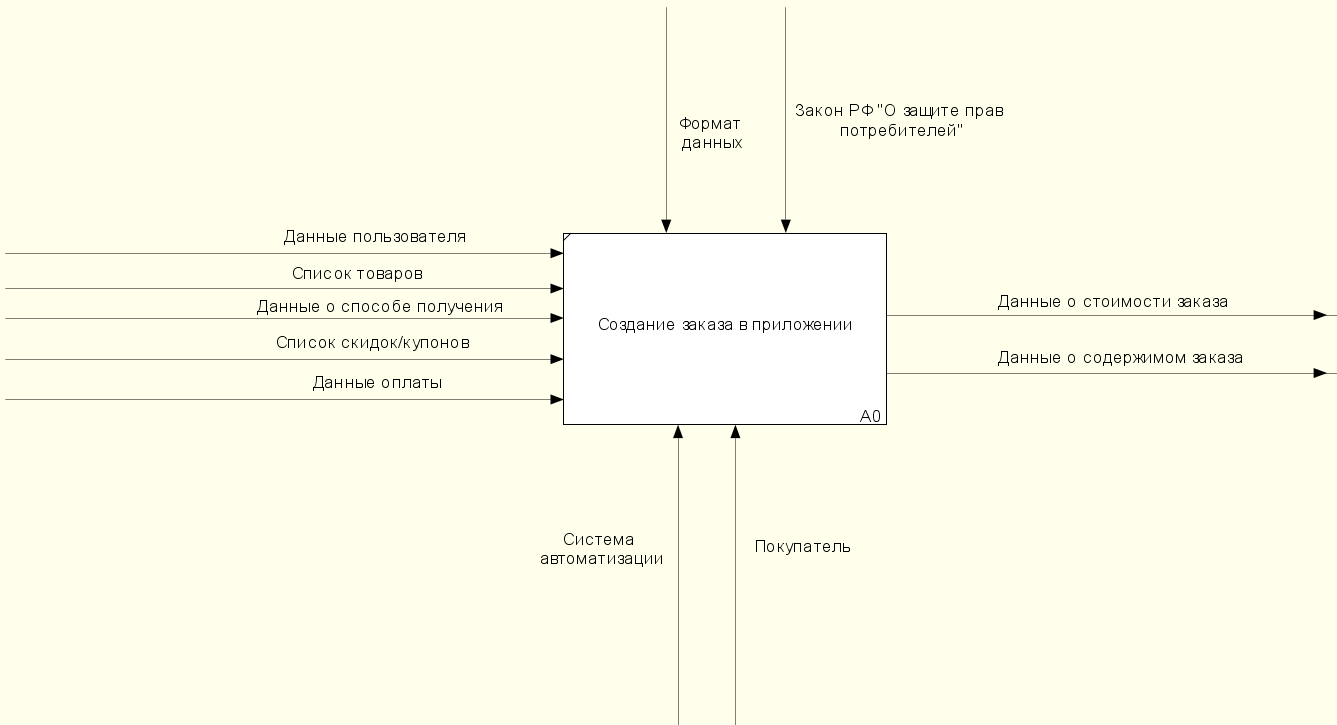
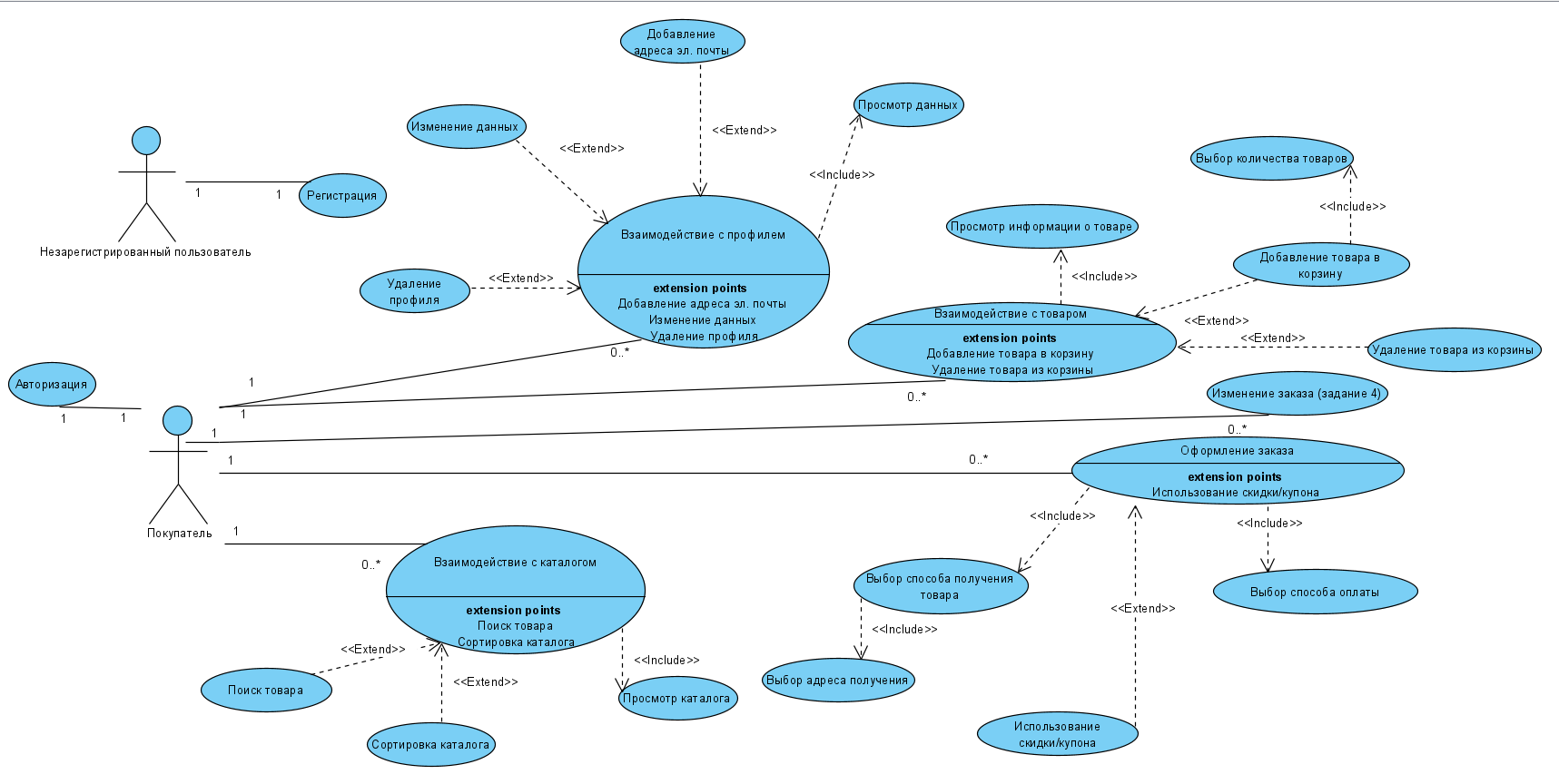
**Дано:**

Мобильное клиент-серверное приложение "Частная пекарня" (приложение для продажи выпечки)

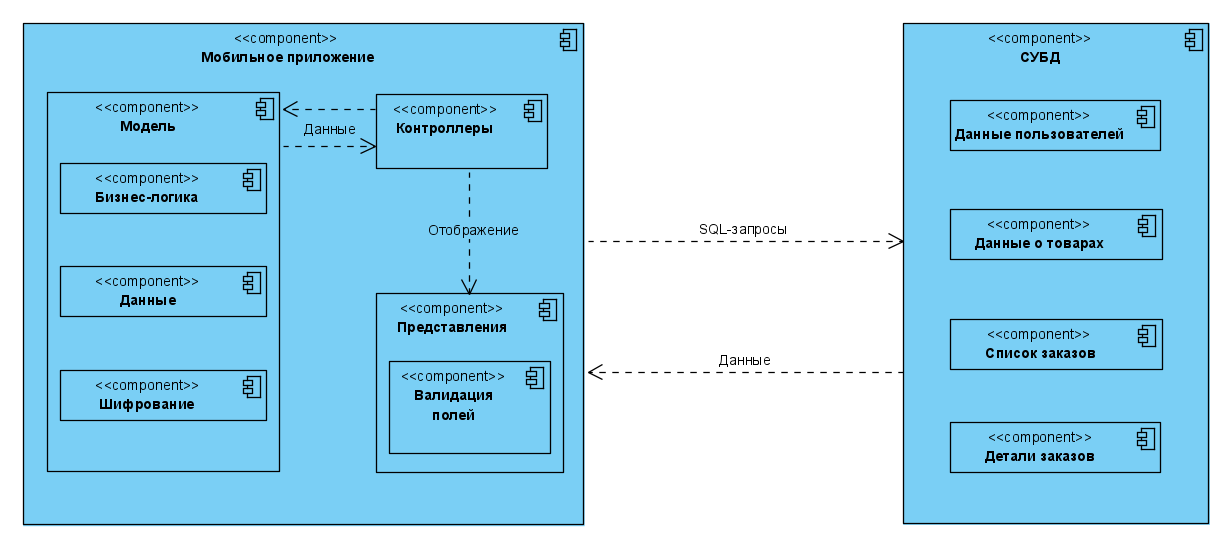
Задание 1. Бизнес-процесс создания заказа (использовать любую удобную нотацию моделирования бизнес-процессов).

Задание 2. Опишите процесс 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, также составить ER-диаграмму сущностей.

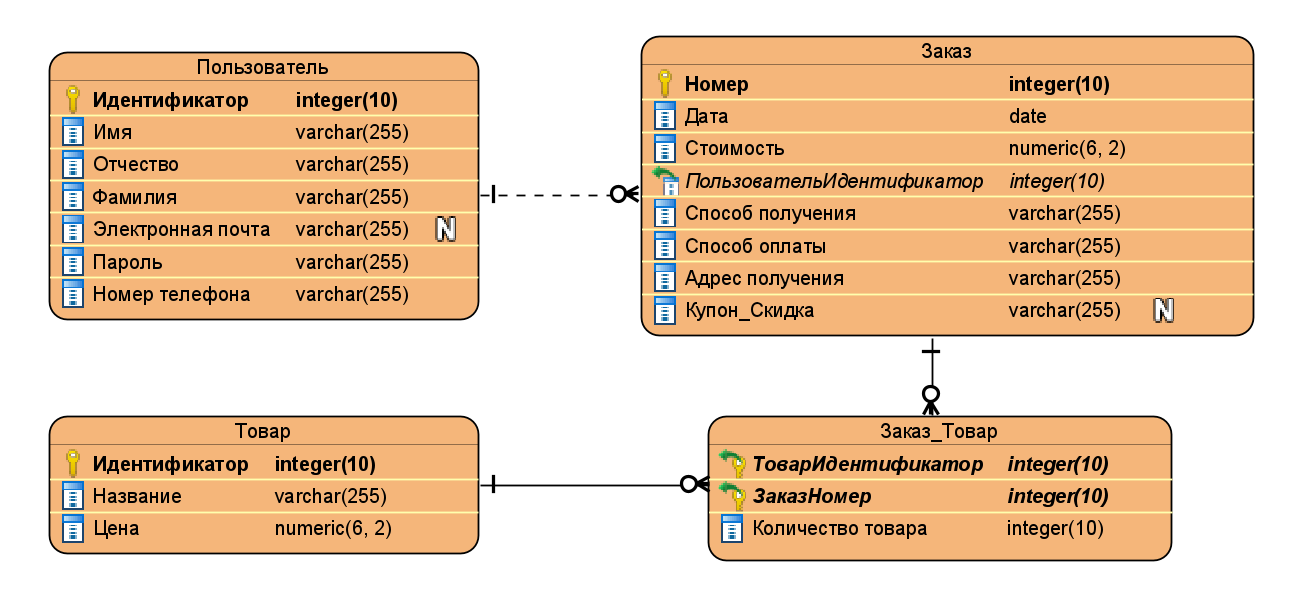
1. Диаграмма вариантов использования



1. Диаграмма компонентов



1. ER-диаграмма



REST-методы

1. Регистрация

— GET-метод для получения формы регистрации.

— POST-метод для добавления данных пользователя в БД. В параметрах могут передаваться переменные типа string для логина/электронной почты, пароля, номера телефона.

2. Авторизация

— GET-метод для получения формы авторизации.

— POST-метод для создания сессии пользователя. В параметрах могут передаваться переменные типа string для логина/электронной почты/номера телефона и пароля.

3. Взаимодействие с каталогом

— GET-метод для получения списка товаров.

— GET-метод для сортировки списка товаров. Тип сортировки зависит от кнопок/пунктов выпадающего списка с типами сортировки.

— GET-метод для поиска товара. В параметрах можно передать содержимое поля для поиска, которое будет сравниваться с названиями товаров в каталоге.

— POST-методы для добавления товара в корзину. В параметрах передается id пользователя и id товара.

— GET-метод для перехода на страницу товара. Передается id товара.

4. Заказ

Оформление:

— GET-метод для получения списка покупаемых товаров. Передается id пользователя.

— GET-метод для получения данных пользователя. Передается id пользователя.

— GET-метод для получения списка скидок/купонов.

— POST-метод для оформления заказа. Передаются id пользователя, список товаров с их стоимостями и количеством, купон и/или скидка (если применены), способ доставки, адрес доставки, способ получения, способ оплаты. Номер и дата заказа генерируются автоматически. Стоимость заказа высчитывается внутри метода. Оплата проходит сразу только если выбран способ оплаты картой.

Редактирование:

— PATCH-метод для редактирования заказа. Передаются номер заказа и объект класса «Заказ» с измененными полями.

Отмена:

— POST-метод с номером заказа в параметрах.

5. Профиль

— PATCH-метод для редактирования данных. Параметры: id пользователя, объект класса «Профиль» с измененными полями.

— GET-метод для получения данных пользователя.

— POST-метод для удаления аккаунта. Параметры: id пользователя.

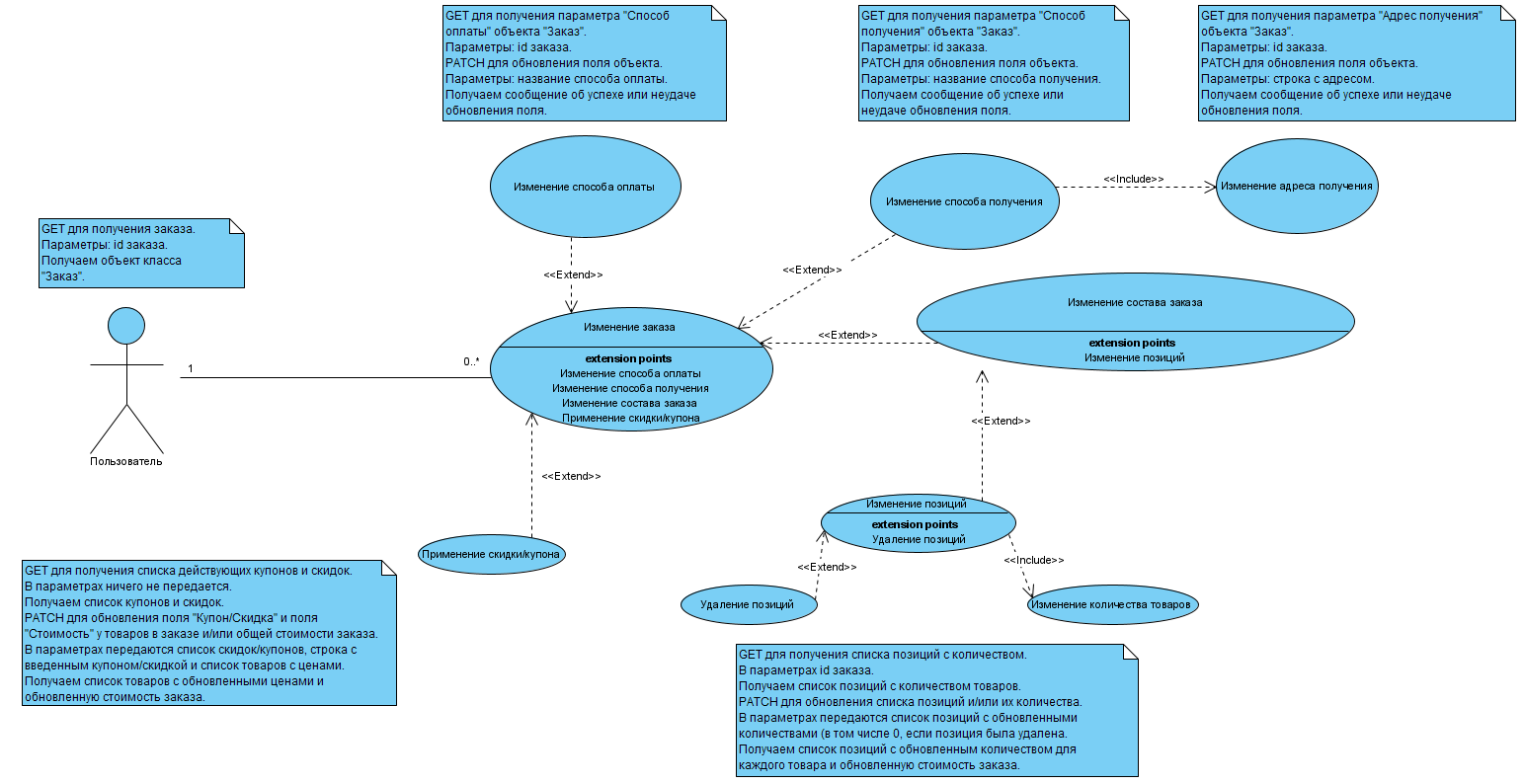
Задание 3.

[Ссылка на прототип (описание интерфейса в нём же).](https://www.figma.com/design/bEx4cSOjAOqKcsnXepmSOQ/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0-%D0%9C%D0%9F-%D0%B4%D0%BB%D1%8F-%D0%BF%D0%B5%D0%BA%D0%B0%D1%80%D0%BD%D0%B8?node-id=0-1&t=Wcyl2qkw4x2wEEh3-1)

Задание 4. Подготовить подробное описание функции редактирования заказа, которую можно было бы использовать в качестве постановки задачи для разработки (помимо текстового описания, использовать UML диаграммы, указать используемые API методы, передаваемые и получаемые параметры, описать процесс хранения информации о покупках пользователя).

API-методы представлены на диаграмме вариантов использования в заметках.

Описание процесса хранения информации о покупках пользователя – см. диаграмму компонентов в задании 2.



5. Запросы:

1) Вывести покупателей с количеством осуществленных покупок

SELECT C.id, C.firstname, C.lastname, COUNT(P.id) AS products\_count FROM clients C JOIN purchases P ON C.id = P.client\_id GROUP BY C.id

2) Вывести общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания

SELECT clients.\*,

(

SELECT SUM(products.price)

FROM purchases

JOIN products ON purchases.product\_id = products.id

WHERE purchases.client\_id = clients.id

) as prices\_sum

FROM clients

ORDER BY prices\_sum DESC;

3) Вывести покупателей, купивших только один товар

SELECT clients ( SELECT COUNT() FROM purchases WHERE purchases.client\_id = clients.id) as products\_count FROM clients HAVING products\_count = 1;