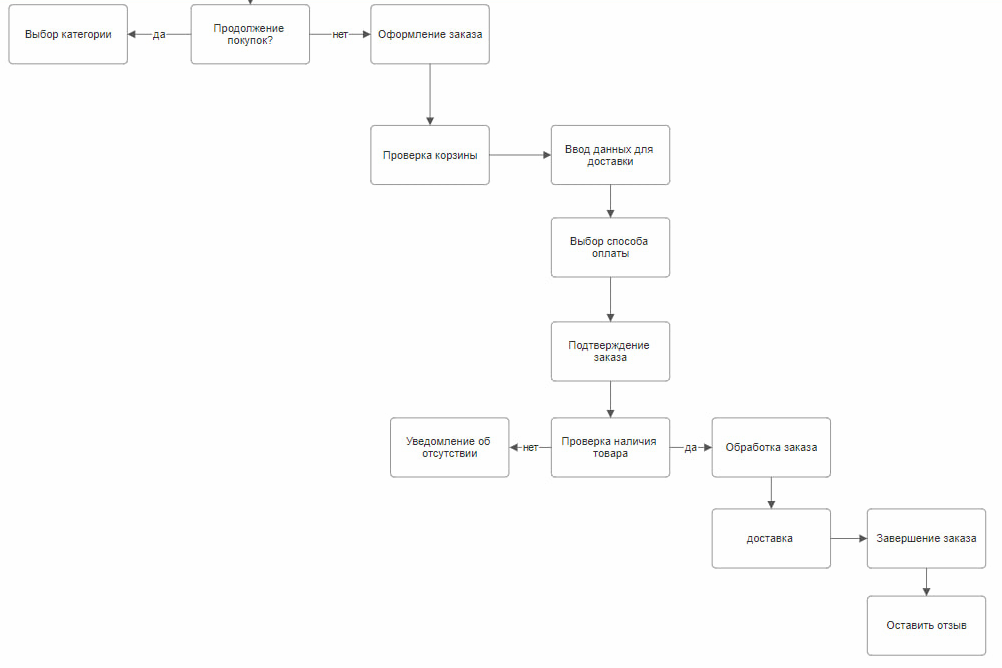
**Задание 1** Бизнес-логику мобильного приложения (использовать любую удобную нотацию моделирования бизнес-процессов).

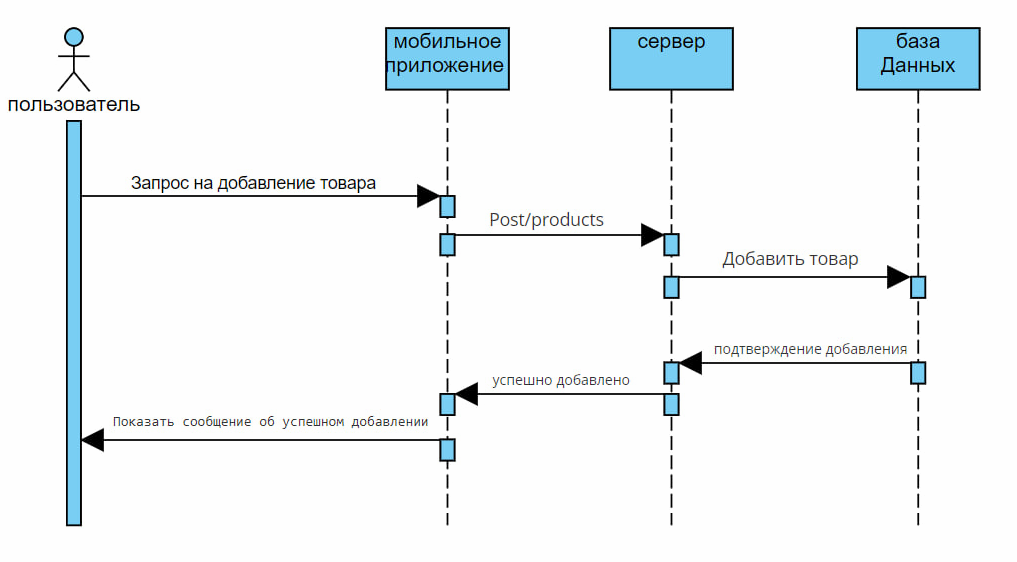
**Задание 2** Основные функции данного приложения (список должен быть составлен в порядке убывания важности функций для пользователя).

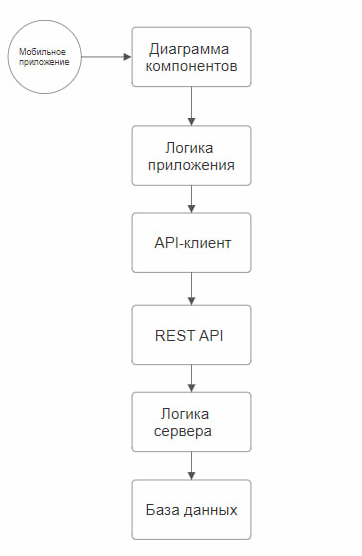
1. **Запуск приложения**:

* Пользователь открывает приложение.
* Проверка, зарегистрирован ли пользователь.

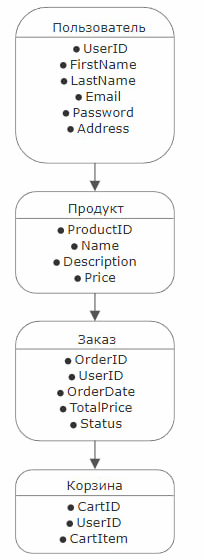
1. **Регистрация и вход**:
   * Если пользователь не зарегистрирован, он выбирает регистрацию.
   * После регистрации, вход в приложение.
2. **Навигация по каталогу**:
   * Выбор категории товара.
   * Просмотр товаров в выбранной категории.
3. **Выбор товара**:
   * Выбор конкретной модели товара.
   * Просмотр деталей товара (описание, отзывы, доступные размеры).
4. **Добавление в корзину**:
   * Добавление товара в корзину.
   * Продолжить покупки или перейти к оформлению заказа.
5. **Оформление заказа**:
   * Проверка содержимого корзины.
   * Пользователь вводит информацию для доставки и выбирает способ оплаты.
   * Подтверждение заказа.
6. **Обработка заказа**:
   * Система проверяет доступность товара на складе.
   * Если товар отсутствует, пользователь получает уведомление.
   * Если товар доступен, заказ обрабатывается и отправляется на склад.
7. **Доставка**:
   * Система назначает курьера.
   * Курьер получает информацию о заказе и доставляет его пользователю.
8. **Завершение заказа**:
   * Пользователь получает уведомление о доставке.
   * Подтверждение получения товара.
   * Оставить отзыв и оценить покупки.

**Задание 3** Процесс синхронизации данных между клиентом и сервером (добавление и удаление списка, наполнение и редактирование списка, покупка/«откупка продукта» и т.д.). Представить все в диаграммах UML, API методах и других представлениях, также составить ER-диаграмму сущностей.

1. Диаграмма последовательности



2. Диаграмма компонентов



3. ER-диаграмма сущностей

### **API Методы:**

**Метод: POST:**

* **Запрос:** {  
   "name": " sneaker store ",  
   "quantity": 5,  
   "price": 622.00  
  }
* **Ответ:** {  
   "item\_id": "12345",  
   "name": " sneaker store ",  
   "quantity": 5,  
   "price": 622.00,  
   "created\_at": "2025-02-13T10:00:00Z"  
  }

**Метод: DELETE:**

* **Запрос:** {  
   "item\_id": "12345"  
  }
* **Ответ:** {  
   "message": "Item deleted successfully"  
  }

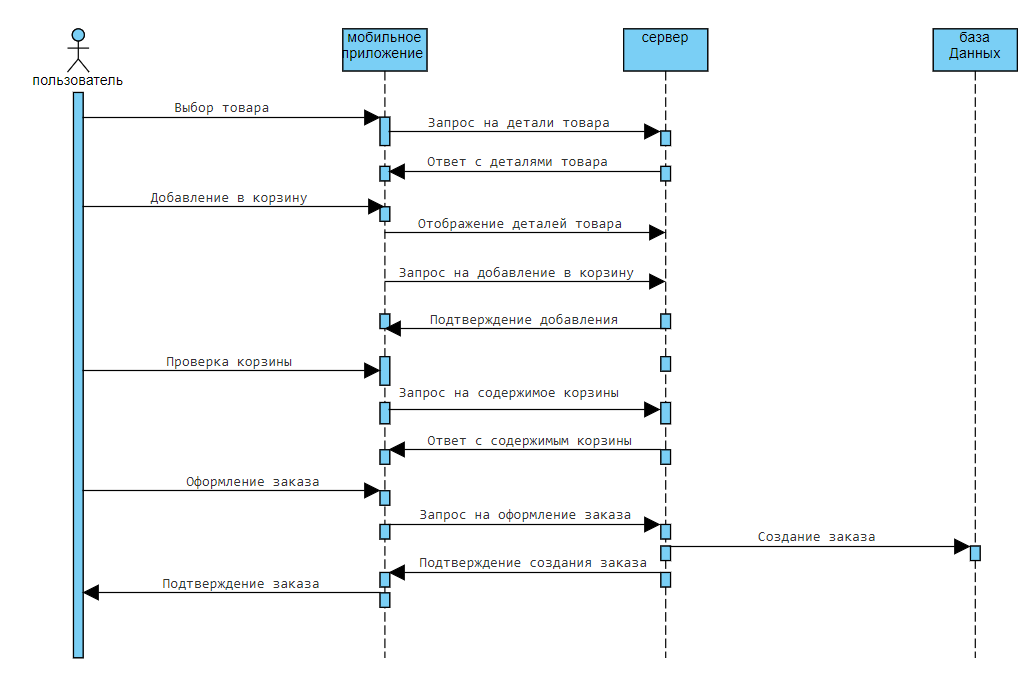
**Метод: PUT:**

* **Запрос:** {  
   "item\_id": "12345",  
   "name": " shoe store ",  
   "quantity": 4,  
   "price": 250.00  
  }
* **Ответ:** {  
   "item\_id": "12345",  
   "name": " shoe store ",  
   "quantity": 4,  
   "price": 250.00  
  }

**Задание 4** Подготовить прототип одного из экранов мобильного приложения и описать пользовательский интерфейс для данного экрана (например, покупка товара).

<https://www.figma.com/design/1m60dPaG33qLkVQPZusAZh/Untitled?node-id=0-1&p=f&t=QGePA5v8vJu5N8Gb-0>

**Задание 5** Подготовить подробное описание функции покупки товара, 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).

**UML Диаграмма**

**Api методы**

* **Метод GET:**

GET /orders/{order\_id}

{  
 "order\_id": "12345",  
 "items": [  
 {  
 "product\_id": "67890",  
 "name": " shoe 1",  
 "quantity": 2,  
 "price": 500  
 }  
 ],  
 "total\_price": 1000  
}

* **Метод PUT:**

PUT /orders/{order\_id}

* **Пример:** {  
   "items": [  
   {  
   "product\_id": "67890",  
   "quantity": 3  
   }  
   ]  
  }
* **Ответ:** {  
   "message": "Order updated successfully"  
  }
* **Метод POST:**

POST /orders

* **Пример:** {  
   "user\_id": "23456",  
   "items": [  
   {  
   "product\_id": "67890",  
   "quantity": 2  
   }  
   ],  
   "total\_price": 1000  
  }
* **Ответ:** {  
   "order\_id": "12345",  
   "message": "Order created successfully"  
  }
* **Метод GET:**

GET /users/{user\_id}/orders

* **Пример:**  
   {  
   "order\_id": "12345",  
   "total\_price": 1000,  
   "created\_at": "2023-10-01T12:00:00Z"  
   },  
   {  
   "order\_id": "12346",  
   "total\_price": 1500,  
   "created\_at": "2023-10-02T15:30:00Z"  
   }  
  ]



**Задание 6** Основные, на ваш взгляд, сложности разработки такого приложения. Вопросы, возникшие при выполнении тестового задания, которые вы бы задали заказчику.

**Сложности разработки**

1. **Интеграция с существующими системами:** может понадобиться связать мобильное приложение с интернет-магазином или системой управления заказами заказчика, чтобы данные синхронизировались.
2. **Адаптация под разные устройства:** важно, чтобы приложение работало на всех мобильных устройствах, независимо от размера экрана и операционной системы.
3. **Безопасность данных:** приложение должно быть защищено от взломов и утечек данных, особенно при обработке платежной информации.
4. **Удобство использования:** приложение должно быть простым и понятным в использовании для пользователей, с легким интерфейсом и быстрой загрузкой данных.
5. **Масштабируемость:** если приложение становится популярным, оно должно выдерживать большие нагрузки и обрабатывать много запросов одновременно.

**Вопросы заказчику**

* Какие функции и возможности должны быть доступны?
* Какие требования безопасности данных должны быть соблюдены в приложении?
* Какие категории товаров должны быть?
* Какие устройства и операционные системы должны поддерживаться приложением?
* Как будет происходить поддержка и обновление приложения?

**Задание 7** . Есть таблицы Books и Authors, где AuthorId табл. Books равно Id табл. Authors. Необходимо написать SQL-запрос, чтобы найти: 1. Общую стоимость книг для каждого автора и отсортировать результат в порядке убывания; 2. Стоимость книг автора превышает 1500; 3. Вывести авторов с количеством книг; 4. Получить автора без книг.

### **Общая стоимость книг для каждого автора и сортировка по убыванию:**

SELECT a.AuthorName, SUM(b.Price) AS TotalPrice FROM Authors a JOIN Books b ON a.Id = b.AuthorId GROUP BY a.AuthorName ORDER BY TotalPrice DESC;

### **Авторы, стоимость книг которых превышает 1500:**

SELECT a.AuthorName, SUM(b.Price) AS TotalPrice FROM Authors a JOIN Books b ON a.Id = b.AuthorId GROUP BY a.AuthorName HAVING TotalPrice > 1500;

### **Вывести авторов с количеством книг:**

SELECT a.AuthorName, COUNT(b.Id) AS BookCount FROM Authors a LEFT JOIN Books b ON a.Id = b.AuthorId GROUP BY a.AuthorName;

### **Получить авторов без книг:**

SELECT a.AuthorName FROM Authors a LEFT JOIN Books b ON a.Id = b.AuthorId WHERE b.Id IS NULL;