Мобильное клиент-серверное приложение "Частная пекарня"

*1)Бизнес-процесс создания заказа*

*Участники:*

1. Клиент - пользователь приложения, который делает заказ.

2. Система - приложение "Частная пекарня", которое обрабатывает заказы.

3. Пекарь - сотрудник пекарни, который готовит выпечку.

*Начало процесса:*

Клиент открывает приложение и выбирает раздел "Заказ".

*Выбор товара:*

Клиент просматривает доступные товары (выпечку) и выбирает желаемые позиции.

Система отображает список доступных товаров с их описанием и ценами.

*Добавление в корзину:*

Клиент добавляет выбранные товары в корзину.

Система обновляет корзину и отображает текущую сумму заказа.

*Проверка заказа:*

Клиент переходит к корзине для проверки заказа.

Система отображает список товаров в корзине, их количество и общую сумму.

Ввод контактной информации:

Клиент вводит контактные данные (имя, телефон, адрес доставки).

Система проверяет корректность введенных данных.

*Выбор способа оплаты:*

Клиент выбирает способ оплаты (онлайн или наложенный платеж).

Система предлагает доступные способы оплаты.

Подтверждение заказа:

Клиент подтверждает заказ.

Система генерирует уникальный номер заказа и отправляет уведомление о подтверждении.

*Уведомление пекаря:*

Система уведомляет пекаря о новом заказе.

Пекарь начинает подготовку заказа.

*Завершение процесса:*

После завершения приготовления, пекарь отмечает заказ как выполненный.

Система обновляет статус заказа и уведомляет клиента о готовности.

Диаграмма последовательности для создания заказа :

Клиент -> Сервер: Создать заказ (POST /orders)

Сервер -> БД: Сохранить заказ

БД -> Сервер: Подтверждение сохранения

Сервер -> Клиент: Успех (200 OK)

Для редактирования заказа:

Клиент -> Сервер: Редактировать заказ (PUT /orders/{id})

Сервер -> БД: Обновить заказ

БД -> Сервер: Подтверждение обновления

Сервер -> Клиент: Успех (200 OK)

Для отмены заказа:

Клиент -> Сервер: Отменить заказ (DELETE /orders/{id})

Сервер -> БД: Удалить заказ

БД -> Сервер: Подтверждение удаления

Сервер -> Клиент: Успех (200 OK)

2)*Диаграмма последовательности (UML Sequence Diagram)*

***1.Описание API методов***

1. Создание заказа

* Метод: POST /api/orders
* Описание: Создает новый заказ.
* Тело запроса:

{

"userId": "string",

"items": [

{

"productId": "string",

"quantity": "integer"

}

],

"deliveryAddress": "string"

}

2. Редактирование заказа

* Метод: PUT /api/orders/{orderId}
* Описание: Обновляет существующий заказ.
* Тело запроса:

{

"items": [

{

"productId": "string",

"quantity": "integer"

}

],

"deliveryAddress": "string"

}

3. Отмена заказа

* Метод: DELETE /api/orders/{orderId}
* Описание: Отменяет существующий заказ.

4. Изменение персональных данных

* Метод: PUT /api/users/{userId}
* Описание: Обновляет данные пользователя.
* Тело запроса:

{

"name": "string",

"email": "string",

"phone": "string"

}

5. Оплата заказа

* Метод: POST /api/payments
* Описание: Обрабатывает платеж за заказ.
* Тело запроса:

{

"orderId": "string",

"paymentMethod": "string",

"amount": "number"

}

***2. ER-диаграмма сущностей***

1. User

* userId: уникальный идентификатор пользователя (PK).
* name: имя пользователя.
* email: электронная почта пользователя.
* phone: телефонный номер пользователя.

2. Order

* orderId: уникальный идентификатор заказа (PK).

• userId: идентификатор пользователя, который сделал заказ (FK).

• status: статус заказа (например, создан, в процессе, отменен, завершен).

• deliveryAddress: адрес доставки.

3. Item

* itemId: уникальный идентификатор элемента заказа (PK).
* orderId: идентификатор заказа, к которому относится элемент (FK).
* productId: идентификатор продукта.
* quantity: количество продукта в заказе.

4. Product

* productId: уникальный идентификатор продукта (PK).
* name: название продукта.
* price: цена продукта.

3*) Прототип экрана "Создание заказа"*

1. Заголовок экрана

• Текст: "Создание заказа"

• Стиль: Большой, жирный шрифт, центрированный в верхней части экрана.

2. Список доступных продуктов

* Элемент: Список (или карточки) продуктов, доступных для заказа.
* Каждая карточка продукта:
* Изображение продукта (например, хлеба или пирога).
* Название продукта.
* Цена продукта.
* Кнопка "Добавить в корзину" с возможностью выбора количества (например, выпадающий список или поле ввода).
* Иконка с информацией о продукте (например, для просмотра описания).

3. Корзина

* Элемент: Панель в нижней части экрана.
* Информация:
* Количество товаров в корзине.
* Общая сумма заказа.
* Кнопка "Перейти к оформлению" (активна только при наличии товаров в корзине).

4. Адрес доставки

* Поле ввода: "Адрес доставки"
* Подсказка: "Введите адрес доставки"
* Возможность автозаполнения адреса на основе текущего местоположения (если разрешено пользователем).
* Кнопка: "Использовать текущее местоположение" (при нажатии запрашивает разрешение на доступ к геолокации).

5. Выбор способа оплаты

* Заголовок: "Выберите способ оплаты"
* Элементы:
* Радиокнопки для выбора способа оплаты (например, "Картой", "Наличными", "Электронный кошелек").
* При выборе "Картой" открывается дополнительное поле для ввода данных карты.

6. Кнопка "Создать заказ"

* Расположение: В нижней части экрана, выделенная цветом.
* Функция: Отправляет запрос на сервер для создания заказа с выбранными продуктами, адресом доставки и способом оплаты.

7. Информационные сообщения

* Элемент: Всплывающее окно или баннер для отображения сообщений об ошибках или успехе (например, "Заказ успешно создан", "Пожалуйста, заполните все поля").

*4)Функция редактирования заказа*

Основные требования

Изменение списка товаров:

* Добавление новых товаров.
* Удаление существующих товаров.
* Изменение количества каждого товара.

2. Обновление контактной информации:

* Изменение имени, телефона и адреса доставки.

3. Обновление метода оплаты:

* Возможность смены метода оплаты (наличные, карта).

4. Подтверждение изменений:

* Пользователь должен подтверждать изменения перед их сохранением.

5. Уведомления:

* Информирование пользователя об успешном обновлении заказа или возникших ошибках

***Диаграмма последовательности***

Пользователь -> Интерфейс: Запрос на редактирование заказа

Интерфейс -> API: GET /api/orders/{orderId}

API -> БД: Получить заказ по ID

БД -> API: Вернуть данные заказа

API -> Интерфейс: Отправить данные заказа

Интерфейс -> Пользователь: Отобразить данные для редактирования

Пользователь -> Интерфейс: Внести изменения

Интерфейс -> API: PUT /api/orders/{orderId} с изменениями

API -> БД: Обновить заказ

БД -> API: Подтверждение обновления

API -> Интерфейс: Успех/Ошибка

Интерфейс -> Пользователь: Уведомление об обновлении

***Используемые API методы***

1. Получение заказа

* Метод: GET /api/orders/{orderId}
* Параметры:
* orderId (параметр URL) — идентификатор заказа.
* Ответ:

{

"id": 1,

"товары": [

{"id": 101, "название": "Торт", "цена": 500, "количество": 1},

{"id": 102, "название": "Кофе", "цена": 150, "количество": 2}

],

"контактныеДанные": {

"имя": "Иван",

"телефон": "+79001234567",

"адрес": "ул. Пушкина, д. 1"

},

"методОплаты": "карта",

"датаСоздания": "2023-10-01T10:00:00Z",

"датаОбновления": "2023-10-01T10:00:00Z"

}

2. Редактирование заказа

* Метод: PUT /api/orders/{orderId}
* Параметры:
* orderId (параметр URL) — идентификатор заказа.
* Тело запроса:

{

"товары": [

{"id": 101, "количество": 2},

{"id": 102, "количество": 1}

],

"контактныеДанные": {

"имя": "Иван",

"телефон": "+79001234567",

"адрес": "ул. Пушкина, д. 2"

},

"методОплаты": "наличные"

}

* Ответ:

{

"success": true,

"message": "Заказ успешно обновлён."

}

3. Удаление товара из заказа

* Метод: DELETE /api/orders/{orderId}/items/{itemId}
* Параметры:
* orderId (параметр URL) — идентификатор заказа.
* itemId (параметр URL) — идентификатор товара.
* Ответ:

{

"success": true,

"message": "Товар успешно удалён из заказа."

}

Процесс хранения информации о покупках пользователя:

1. База данных:

* Заказы хранятся в таблице orders, которая содержит следующие поля:
* id: уникальный идентификатор заказа.
* userId: идентификатор пользователя, который создал заказ.
* товары: JSON-поле, содержащее список товаров с их количеством и ценой.
* контактныеДанные: JSON-поле с информацией о пользователе.
* методОплаты: строка, указывающая способ оплаты.
* датаСоздания: дата и время создания заказа.
* датаОбновления: дата и время последнего обновления заказа.

2. Связь с пользователем:

* Каждому заказу присваивается идентификатор пользователя (userId), чтобы отслеживать заказы конкретного клиента.

3. Обновление информации:

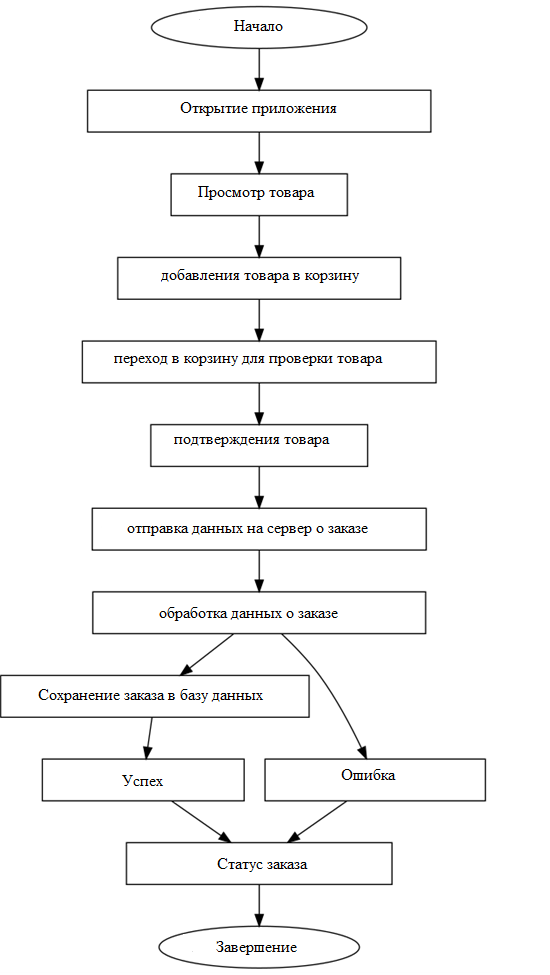
* При редактировании заказа информация о товарах и контактных данных обновляется в базе данных с помощью API метода PUT. Обновление происходит по уникальному идентификатору заказа.

4. Логирование изменений:

* Все изменения фиксируются в отдельной таблице order\_updates, где хранятся старые и новые значения для отслеживания истории изменений. Это может включать поля:
* updateId: уникальный идентификатор записи об обновлении.
* orderId: идентификатор измененного заказа.
* oldData: старые данные заказа в формате JSON.
* newData: новые данные заказа в формате JSON.
* updatedAt: дата и время изменения.

5. Безопасность данных:

* Все запросы к API должны быть защищены аутентификацией (например, с использованием токенов), чтобы предотвратить несанкционированный доступ к данным заказов.



5)  *Реляционная модель данных*

-- 1. Покупатели с количеством покупок

SELECT

p.customer\_id,

COUNT(o.order\_id) AS purchase\_count,

-- 2. Общая стоимость товаров для каждого покупателя

SUM(oi.quantity \* oi.price) AS total\_spent

FROM

customers p

JOIN

orders o ON p.customer\_id = o.customer\_id

JOIN

order\_items oi ON o.order\_id = oi.order\_id

GROUP BY

p.customer\_id

-- 3. Покупатели, которые купили только один товар

HAVING

COUNT(DISTINCT oi.product\_id) = 1

ORDER BY

total\_spent DESC;