Дано:

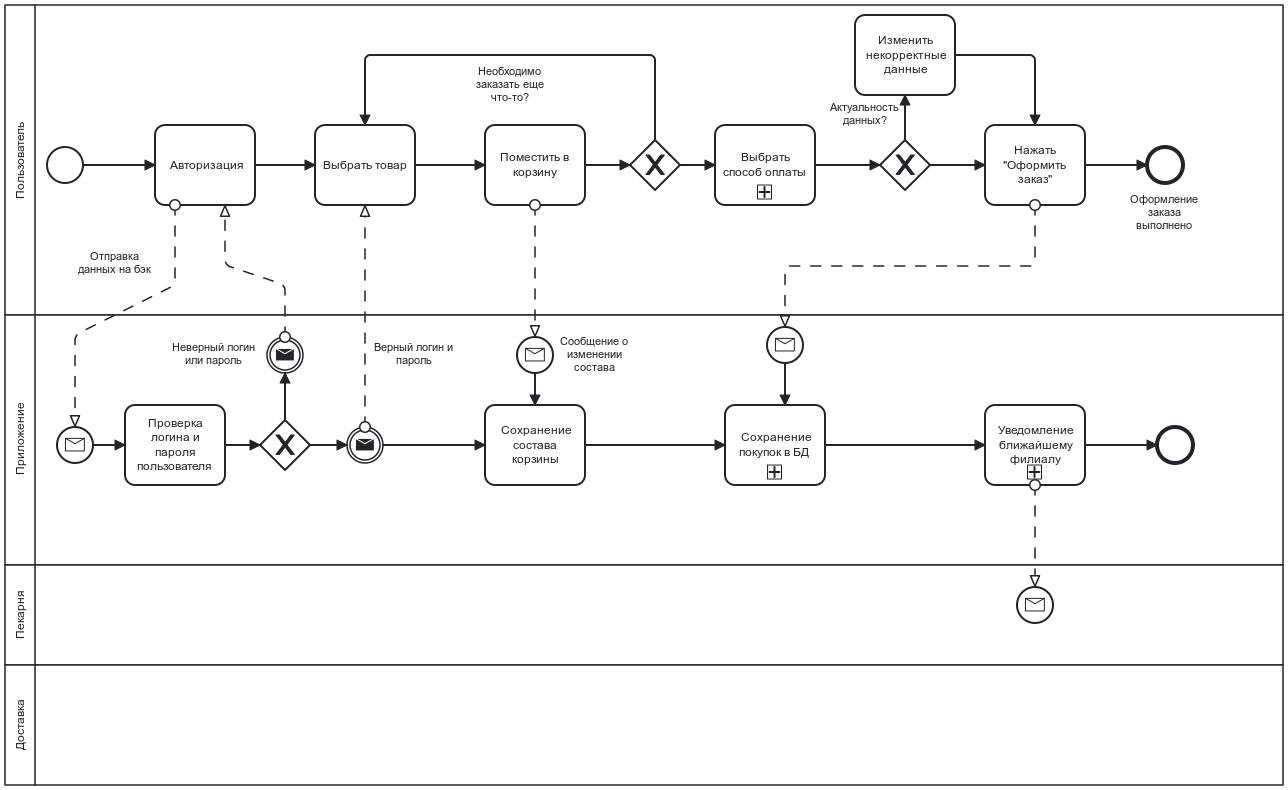
Мобильное клиент-серверное приложение "Частная кофейня" (приложение для продажи кофе и выпечки)

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

BPMN (Business Process Model and Notation — Модель и нотация бизнес-процессов) — это стандартизированный графический язык, который позволяет описывать и документировать бизнес-процессы в виде диаграмм.

Разработанный Object Management Group (OMG), BPMN предназначен для использования всеми заинтересованными сторонами, включая бизнес-аналитиков, разработчиков ПО, менеджеров проектов и любых других лиц, участвующих в определении и анализе бизнес-процессов.

BPMN обеспечивает единый визуальный язык, который помогает улучшить коммуникацию между различными участниками проекта и упрощает процесс моделирования бизнес-процессов.



1. 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.

Процесс синхронизации данных между клиентом и сервером в контексте создания, редактирования и отмены заказа, а также обработки персональных данных и оплаты заказа, может быть описан следующим образом:

1. ”Создание заказа”:

- Клиент отправляет запрос на сервер с деталями заказа через API метод `POST /orders`.

- Сервер обрабатывает запрос, создает новый заказ в базе данных и возвращает клиенту подтверждение с уникальным идентификатором заказа.

2. ”Редактирование заказа”:

- Клиент отправляет изменения в заказе на сервер через API метод `PUT /orders/{orderId}` с указанием идентификатора заказа.

- Сервер обновляет информацию о заказе в базе данных и отправляет клиенту подтверждение об успешном обновлении.

3. ”Отмена заказа”:

- Клиент запрашивает отмену заказа, используя API метод `DELETE /orders/{orderId}`.

- Сервер обрабатывает запрос на отмену, обновляет статус заказа в базе данных и информирует клиента об успешной отмене.

4. “Изменение персональных данных”:

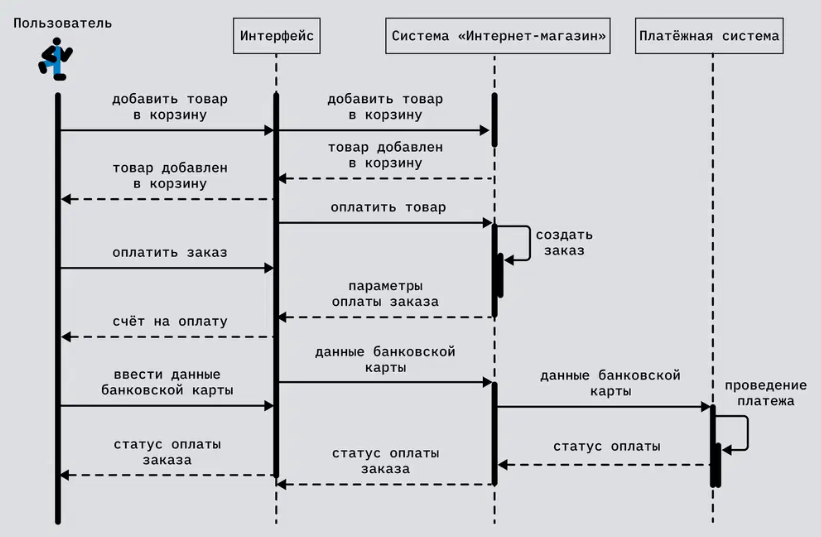
- Клиент отправляет запрос на обновление персональных данных через API метод `PATCH /users/{userId}`.

- Сервер обновляет персональные данные пользователя в базе данных и подтверждает клиенту успешное обновление.

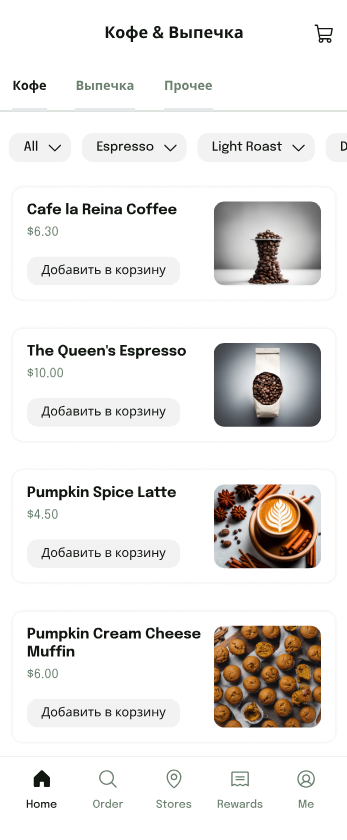
5. “Оплата заказа”:

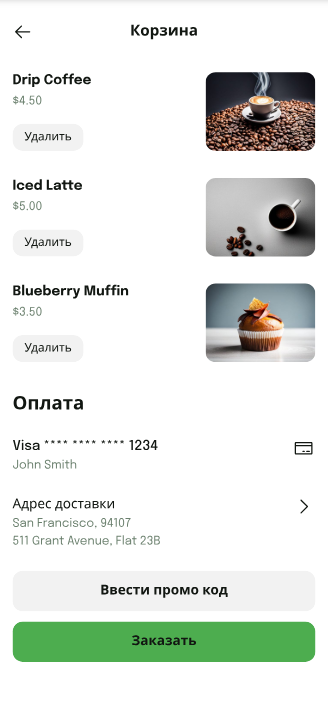
- Клиент отправляет данные для оплаты через безопасное соединение, используя API метод `POST /payments`.

- Сервер обрабатывает платеж, регистрирует его в системе и отправляет клиенту подтверждение об успешной оплате.



1. Подготовить прототип одного из экранов данного мобильного приложения и описать пользовательский интерфейс для данного экрана (например, создание заказа).





Экран "Создание заказа"

1. “Верхняя панель”:

- Иконка корзины в правом верхнем углу (показывающая количество добавленных товаров).

2. “Выбор напитков”:

- Изображении с популярными товарами для быстрого выбора.

- Под каждым изображением - название напитка и кнопка "Добавить в корзину".

3. “Выбор карты”:

- Поле для ввода карты с автозаполнением и возможностью выбора из сохраненных карт.

4. “Адрес доставки”:

- Поле для ввода адреса с автозаполнением и возможностью выбора из сохраненных адресов.

5. “Подтверждение заказа”:

- Кнопка "Оформить заказ", ведущая к экрану оплаты.

6. “Ввод промо кода”:

- Поле "Ввод промо кода" позволяет получить скидку от партнеров.

7. “Нижняя панель”:

- Иконка "Главная" для возврата на главный экран.

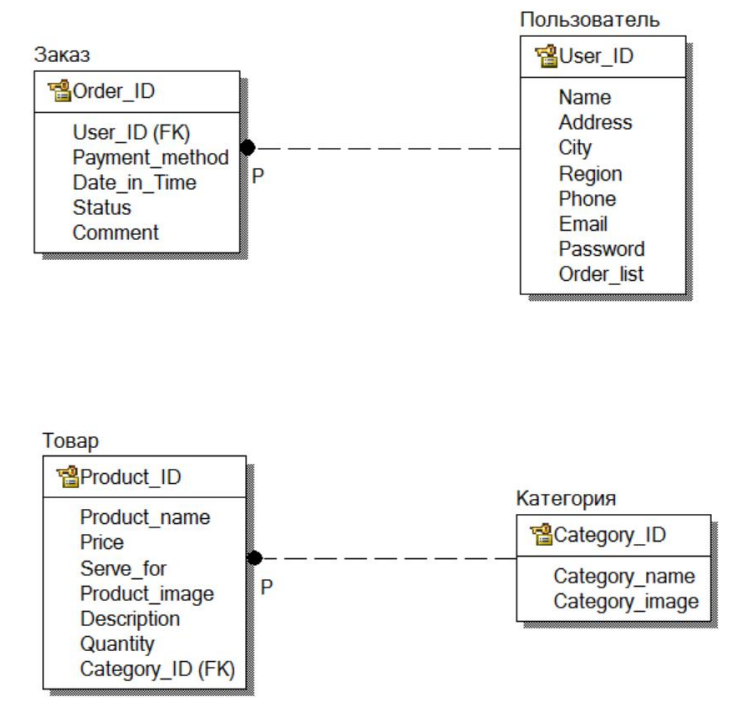
- Иконка "Поиск" для доступа к большему кол-ву товаров.

- Иконка "Пункты (Филиалы)" для поиска филиалов компании.

- Иконка "Отзывы" для общения и обратной связи.

- Иконка "Профиль" для входа в личный кабинет пользователя.

1. 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).



Функция позволяет пользователям изменять параметры своих заказов через личный кабинет на сайте. Это включает в себя изменение количества товаров, выбора товаров, способа и адреса доставки, а также времени доставки.

API Методы:

GET /orders/{orderId}/edit

Описание: Получение данных о заказе для редактирования.

Параметры:

orderId (path) - ID заказа.

Ответ: Объект заказа с возможными параметрами для редактирования.

PUT /orders/{orderId}

Описание: Обновление параметров заказа.

Параметры:

orderId (path) - ID заказа.

orderData (body) - Объект с измененными данными заказа.

Ответ: Подтверждение об успешном обновлении заказа.

Параметры передачи:

orderData:

quantity (int) - Количество товаров.

items (array) - Список товаров.

deliveryMethod (string) - Способ доставки.

deliveryAddress (string) - Адрес доставки.

deliveryTime (string) - Время доставки.

Процесс хранения информации о покупках пользователя:

* Информация о заказах пользователя хранится в централизованной базе данных.
* При каждом изменении заказа создается запись в истории заказов, содержащая детали изменений.
* Для обеспечения безопасности, все транзакции с данными защищены и используются методы шифрования.

Пример использования:

PUT /orders/12345

{

"quantity": 2,

"items": [

{"id": "item1", "name": "Товар 1"},

{"id": "item2", "name": "Товар 2"}

],

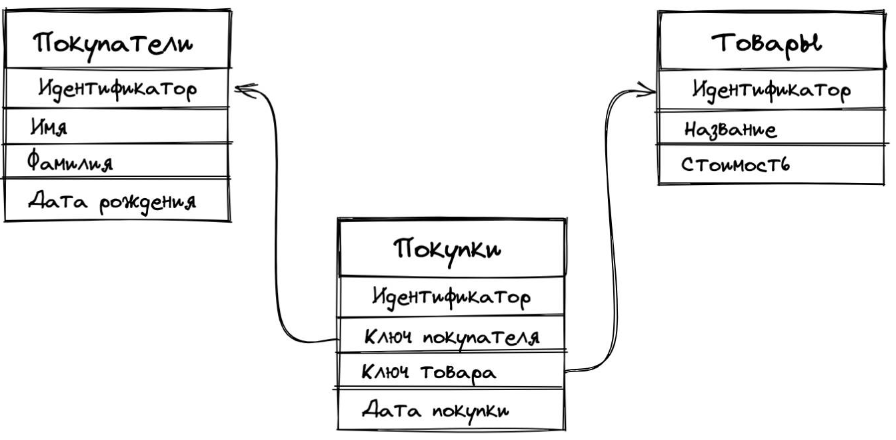
"deliveryMethod": "express",

"deliveryAddress": "123 Main St, Город, Страна",

"deliveryTime": "2024-07-10T14:00:00Z"

}

1. Перед вами реляционная модель данных. Необходимо написать SQL-запросы: вывести покупателей с количеством осуществленных покупок, общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания, получить покупателей, купивших только один товар.



-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

FROM Покупки

JOIN Покупатели ON Покупки.Ключ\_покупателя = Покупатели.Идентификатор

GROUP BY Покупатели.Идентификатор, Покупатели.Имя, Покупатели.Фамилия

HAVING COUNT(Покупки.Идентификатор) = 1;