ТЕСТОВОЕ ЗАДАНИЕ НА ПОЗИЦИЮ СИСТЕМНЫЙ АНАЛИТИК

1 БАЗЫ ДАННЫХ – ТЕСТ

Для приведенных ниже вопросов выберите верные утверждения. Верным может быть одно или несколько утверждений (варианты, выделенные **жирным,** являются верными).

1	Содержит ли какую-то информацию таблица, в которой нет полей?	1. Содержит информацию о структуре БД 2. Не содержит никакой информации **3. Таблица без полей существовать не может** 4. Содержит информацию о будущих записях
2	В записи файла реляционной БД может содержаться:	1. Исключительно однородная информация; (данные только одного типа); 2. Только текстовая информация; 3. Только логические величины; **4. Неоднородная информация (данные разных типов);** 5. Исключительно числовая информация.
3	Чем первичный ключ отличается от внешнего ключа?	1. Первичный ключ всегда состоит из множества столбцов, а внешний ключ состоит из одного столбца; **2. Значения первичного ключа всегда должны быть уникальными и не могут быть null, значения внешнего ключа могут повторяться;** 3. Внешний ключ является идентификатором строки, а первичный ключ используется для связи между таблиц; **4. Первичный ключ является идентификатором для строки, а внешний ключ используется для связывания таблиц.**
4	В какой нормальной форме говорится о том, что все атрибуты зависят от первичного ключа, а не от его части?	1. 1НФ; **2. 2НФ;** 3. 3НФ; 4. 4НФ.
5	В каком порядке в СУБД выполняются операторы SELECT, FROM, GROUP BY?	1. Сначала SELECT, потом FROM и только потом GROUP BY; 2. Сначала GROUP BY, потом SELECT и только потом FROM; 3. Сначала FROM, потом SELECT и только потом GROUP BY; **4. Сначала FROM, потом GROUP BY и только потом SELECT.**

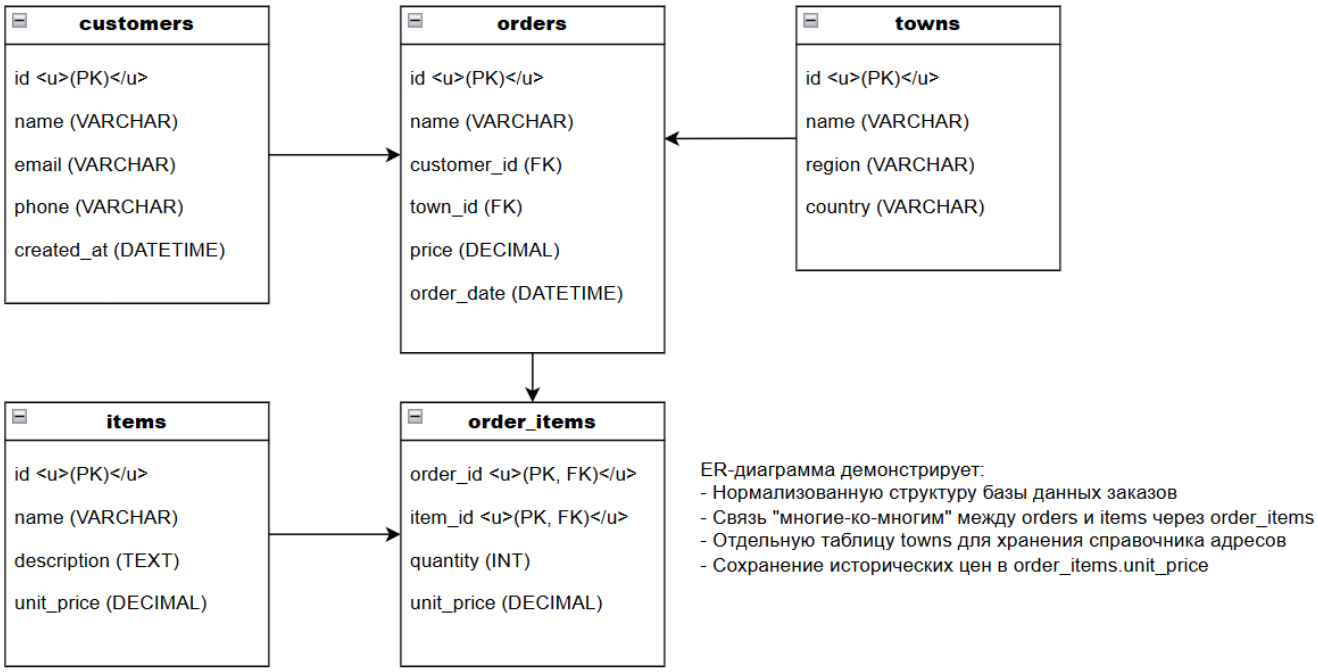
<table>
<tr><td>6</td><td>Чем отличается оператор WHERE от HAVING?</td><td>1. Оператор WHERE применяется для фильтрации групп, а HAVING - для фильтрации отдельных строк;
2. Оператор HAVING применяется для фильтрации групп, а WHERE - для фильтрации отдельных строк;
3. HAVING работает только с агрегатными функциями, а WHERE может работать с любыми типами выражений;
4. WHERE может использоваться для фильтрации по любому полю или выражению, а HAVING - только для фильтрации по выражению в списке выбора или агрегатной функции;
5. HAVING всегда используется после GROUP BY, а WHERE может использоваться до или после GROUP BY.</td></tr>
<tr><td>7</td><td>Какой результат покажет выполнение операторов SELECT COUNT (*)?</td><td>1. Число строк таблицы, указанной во FROM, не включая значение;
2. Число строк таблицы, указанной во FROM, где ячейка содержит символ;
3. Число строк таблицы, указанной во FROM, включая значение;
4. Сумма строк таблицы, указанной во FROM, где ячейка содержит символ.</td></tr>
<tr><td>8</td><td>В таблице «Animals» базы данных зоопарка содержится информация обо всех обитающих там животных, в том числе о лисах: red fox, grey fox, little fox. Напишите запрос, возвращающий информацию о возрасте лис</td><td>1. SELECT age FROM Animals WHERE Animal LIKE “%fox”;
2. SELECT age FROM %Fox.Animals;
3. SELECT age FROM Animals WHERE Animal = fox;
4. SELECT %fox age FROM Animals.</td></tr>
<tr><td>9</td><td>Чем отличается DELETE от TRUNCATE?</td><td>1. DELETE и TRUNCATE – это одно и то же;
2. DELETE используется для удаления одной или нескольких строк из таблицы, а TRUNCATE используется для удаления всех строк из таблицы;
3. DELETE может использовать условие WHERE, а TRUNCATE всегда удаляет все записи из таблицы;
4. DELETE удаляет данные из таблицы, а TRUNCATE удаляет саму таблицу.</td></tr>
<tr><td>10</td><td>Дана таблица:
<table><tr><td>COLOR</td></tr><tr><td>BLUE</td></tr><tr><td>RED</td></tr><tr><td>RED</td></tr></table>Каким будет результат запроса?
SELECT COUNT (DISTINCT color) FROM Table</td><td>1. BLUE, RED, NULL;
2. 3;
3. 1,2,4;
4. 2.</td></tr>
</table>

2 БАЗЫ ДАННЫХ – ER

В базе данных есть таблица заказов - orders. В ней есть поля: id (идентификатор заказа), name (название заказа), town (адрес доставки заказа), price (цена заказа), customer_id (идентификатор покупателя). Также есть таблицы: towns (справочник адресов), items (товары), customers (покупатели). Известно, что между orders и items предполагается связь многие-ко-многим.

Что нужно сделать:

Спроектируйте ER-диаграмму с учетом этих вводных. Состав полей таблиц укажите на свое усмотрение с учетом условия, но для таблицы orders учтите те поля, что указаны.



3 ИНТЕГРАЦИИ

Представим, что Вы работаете аналитиком и проектируете работу приложения интернет-магазина. Вам нужно заложить и спроектировать следующий сценарий: отображение витрины товаров (список товаров с кратким описанием), переход с витрины на экран с детальным описанием конкретного товара, добавление товара в корзину.

Что нужно сделать:

1. Спроектируйте REST API, которые нужны для реализации описанного сценария. Решение должно включать описание запросов и описание или пример ответа для каждого из запросов в формате JSON. Способ описания - на ваше усмотрение. По составу полей товара можете ориентироваться на любой известный интернет-магазин.

Базовая информация:

- Базовый URL: https://api.shop.com/v1;
- Формат данных: JSON;
- Аутентификация: Bearer Token (для защищенных endpoints).

3.1 Получение витрины товаров

Endpoint: GET /products

Параметры запроса:

- category (опционально) - фильтр по категории;
- page (опционально) - номер страницы (пагинация);
- limit (опционально) - количество товаров на странице;
- sort (опционально) - сортировка (price_asc, price_desc, popular).

Пример запроса:

GET /products?category=electronics&page=1&limit=20&sort=popular
Authorization: Bearer {token}

Пример ответа:


```
{
  "status": "success",
  "data": {
    "products": [
      {
        "id": "prod_001",
        "name": "Смартфон Samsung Galaxy S23",
        "shortDescription": "Флагманский смартфон с камерой 200 МП",
        "price": 79999,
        "currency": "RUB",
        "imageUrl": "/images/products/s23.jpg",
        "rating": 4.8,
        "reviewCount": 124,
        "inStock": true,
        "category": "electronics"
      },
      {
        "id": "prod_002",
        "name": "Наушники Sony WH-1000XM4",
        "shortDescription": "Беспроводные наушники с шумоподавлением",
        "price": 24999,
        "currency": "RUB",
        "imageUrl": "/images/products/sony-xm4.jpg",
        "rating": 4.9,
        "reviewCount": 89,
        "inStock": true,
        "category": "electronics"
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 5,
      "totalProducts": 95,
      "productsPerPage": 20
    }
  }
}
```


3.2 Получение детальной информации о товаре

Endpoint: GET /products/{productId}
Пример запроса:

GET /products/prod_001
Authorization: Bearer {token}

Пример ответа:


```
{
  "status": "success",
  "data": {
    "product": {
      "id": "prod_001",
      "name": "Смартфон Samsung Galaxy S23",
      "fullDescription": "Флагманский смартфон с революционной камерой 200 МП...",
      "price": 79999,
      "currency": "RUB",
      "images": [
        "/images/products/s23-1.jpg",
        "/images/products/s23-2.jpg",
        "/images/products/s23-3.jpg"
      ],
      "specifications": {
        "display": "6.1 дюйма, Dynamic AMOLED 2X",
        "processor": "Snapdragon 8 Gen 2",
        "memory": "8GB RAM, 256GB ROM",
        "camera": "200MP + 10MP + 12MP",
        "battery": "3900 mAh"
      },
      "rating": 4.8,
      "reviewCount": 124,
      "inStock": true,
      "stockQuantity": 15,
      "category": "electronics",
      "brand": "Samsung",
      "warranty": "24 месяца"
    }
  }
}
```


3.3 Добавление товара в корзину

Endpoint: POST /cart/items
Тело запроса:


```
{
  "productId": "prod_001",
  "quantity": 1
}
```


Пример запроса:


```
POST /cart/items
Authorization: Bearer {token}
Content-Type: application/json

{
  "productId": "prod_001",
  "quantity": 1
}
```

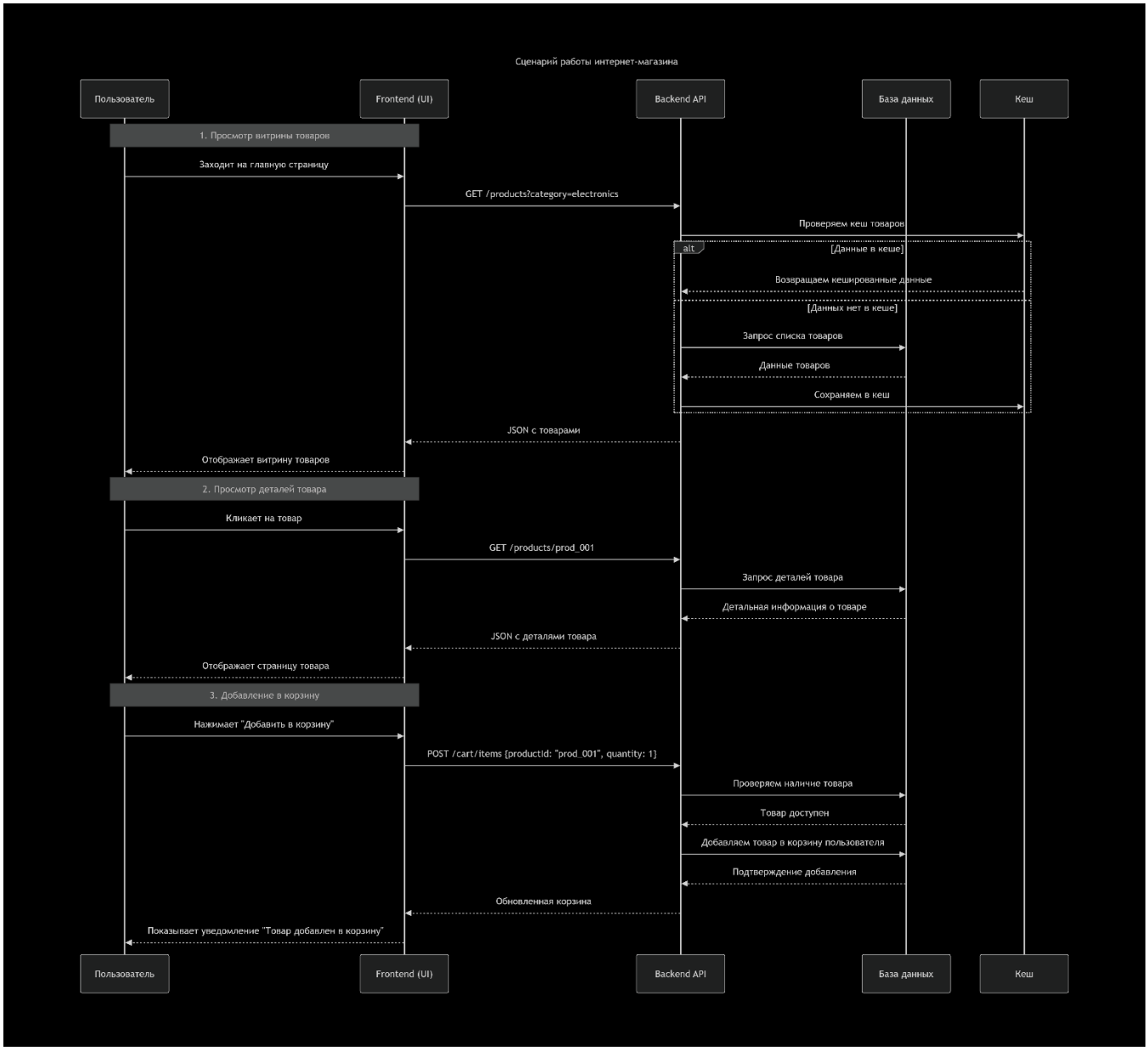
Пример ответа:


```
{
  "status": "success",
  "data": {
    "cart": {
      "id": "cart_12345",
      "userId": "user_001",
      "items": [
        {
          "productId": "prod_001",
          "name": "Смартфон Samsung Galaxy S23",
          "price": 79999,
          "quantity": 1,
          "imageUrl": "/images/products/s23.jpg",
          "subtotal": 79999
        }
      ],
      "totalItems": 1,
      "totalPrice": 79999,
      "currency": "RUB"
    }
  }
}
```


Дополнительные endpoints:

- Получение корзины: GET /cart;
- Обновление количества: PUT /cart/items/{productId};
- Удаление из корзины: DELETE /cart/items/{productId}.

3.4 Постройте Sequence UML диаграмму для этого сценария



Основные компоненты системы:

- Пользователь - взаимодействует с интерфейсом;
- Frontend - отображает UI и обрабатывает действия пользователя;
- Backend API - обрабатывает бизнес-логику;
- База данных - хранит информацию о товарах, корзинах, пользователях;
- Кеш - ускоряет доступ к часто запрашиваемым данным.

4 АЛГОРИТМИЧЕСКОЕ МЫШЛЕНИЕ

Возьмем в качестве примера банковское мобильное приложение. Исходные условия: у вас в руке смартфон, на котором установлено банковское приложение (телефон выключен).

Что нужно сделать: Используя любую нотацию, опишите в виде диаграммы процесс, в рамках которого Вы, используя банковское приложение, пополните баланс своего телефона на 100Р. В качестве примера можете использовать любое мобильное банковское приложение, что у Вас есть.

1) Подготовка устройства

– Включение телефона – нажать и удерживать кнопку питания;
– Разблокировка – ввести PIN-код/графический ключ или использовать биометрию;
– Поиск приложения – найти иконку банковского приложения на рабочем столе или в меню.

2) Аутентификация в приложении

– Запуск приложения – тап по иконке банка;
– Вход в систему:
  - Ввод PIN-кода приложения ИЛИ
  - Сканирование отпечатка пальца ИЛИ
  - Распознавание лица.

3) Навигация к функции пополнения

– Поиск раздела платежей - переход в меню «Платежи», «Переводы» или «Услуги»;
– Выбор категории - тап на «Мобильная связь», «Телефон» или «Пополнение счета»;
– Выбор оператора - из списка выбрать своего оператора связи (МТС, Билайн, МегаФон, Tele2).

4) Ввод данных платежа

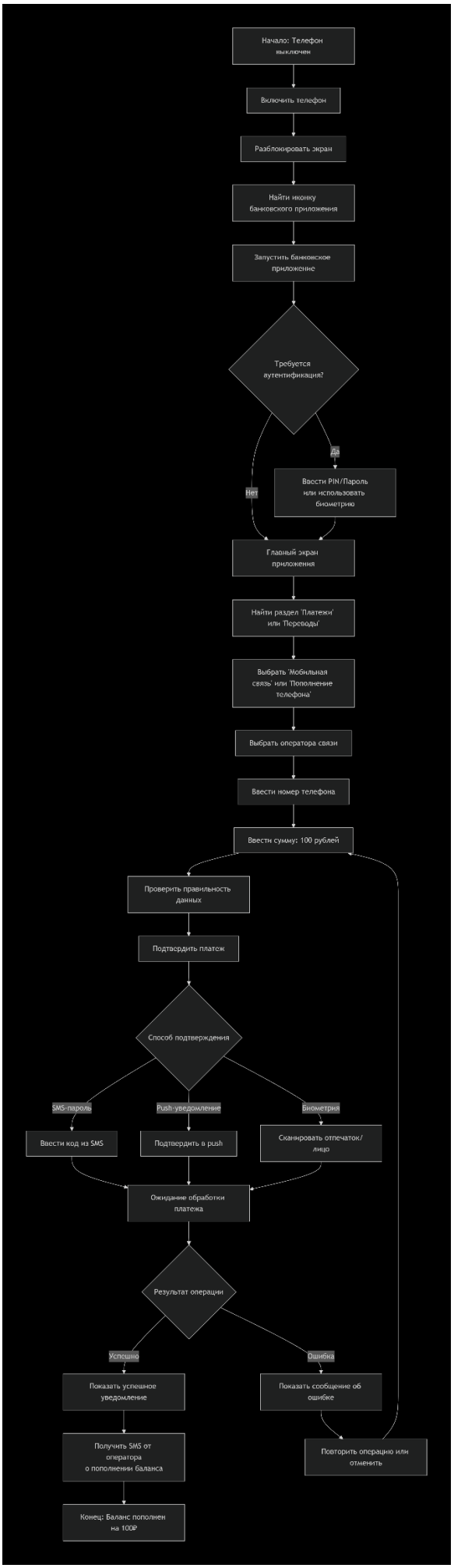
– Ввод номера телефона:
  - Автоподстановка своего номера ИЛИ
  - Ручной ввод номера;
– Указание суммы - ввод «100» рублей;
– Проверка данных - визуальная проверка номера и суммы.

5) Подтверждение операции

– Подтверждение платежа - нажать кнопку «Оплатить», «Продолжить»
– Дополнительная аутентификация:
  - Ввод SMS-кода ИЛИ
  - Подтверждение через push-уведомление ИЛИ
  - Биометрическая аутентификация.

6) Завершение операции

– Ожидание обработки - отображение индикатора загрузки;
– Получение результата:
  - Успех: «Платеж выполнен успешно»;
  - Ошибка: описание проблемы и рекомендации;
  - Подтверждение от оператора - получение SMS о пополнении баланса.


Начало: Телефон выключен
Включить телефон
Разблокировать экран
Найти иконку банковского приложения
Запустить банковское приложение
Требуется аутентификация?
Да
Нет
Ввести PIN/Пароль или использовать биометрию
Главный экран приложения
Найти раздел 'Платежи' или 'Переводы'
Выбрать 'Мобильная связь' или 'Пополнение телефона'
Выбрать оператора связи
Ввести номер телефона
Ввести сумму: 100 рублей
Проверить правильность данных
Подтвердить платеж
Способ подтверждения
SMS-пароль
Push-уведомление
Биометрия
Ввести код из SMS
Подтвердить в push
Сканировать отпечаток/лицо
Ожидание обработки платежа
Результат операции
Успешно
Ошибка
Показать успешное уведомление
Показать сообщение об ошибке
Получить SMS от оператора о пополнении баланса
Повторить операцию или отменить
Конец: Баланс пополнен на 100₽