Практические задания для Славика

Задание 1: Уточнение требований

Представь, что ты аналитик и тебе прилетело требование от бизнеса реализовать в интернет-банке оповещение клиентов о списаниях и зачислениях по счету. Твоя задача - раскопать требования, чтобы можно было приступить к проектированию. Какие вопросы ты бы задал?

Задание 2: Построение BPMN-диаграммы бизнес-процесса

Контекст задачи

Твоя задача — проанализировать и визуализировать процесс обработки клиентского запроса на оформление кредита. Процесс охватывает взаимодействие клиента, сотрудников банка и автоматизированных систем.

Описание процесса

1. **Инициация запроса**
   Клиент обращается в банк через один из каналов:
   - Онлайн-форма на сайте/в мобильном приложении
   - Личное посещение офиса банка
2. **Проверка данных клиента**
   - Если клиент обращается онлайн, система автоматически проверяет заполненные данные (валидация полей).
   - Если клиент в офисе, сотрудник вводит данные в систему и запускает процесс проверки.
   - На этом этапе проверяется:
     - Наличие клиента в базе банка.
     - Соответствие заполненных данных требованиям банка.

   Результат проверки:

   - Если данные некорректны или не соответствуют требованиям, клиенту отправляется уведомление (в приложении или устно в офисе).
   - Если данные корректны, процесс продолжается.

3. **Оценка кредитоспособности клиента**
   - Система запрашивает данные у внешних сервисов:
     - Кредитное бюро (проверка кредитной истории).
     - Налоговая служба (проверка доходов, если требуется).
   - Параллельно проверяется текущая задолженность клиента перед банком.
   - На основе собранных данных система рассчитывает скоринговый балл.
4. **Принятие решения**
   - Если скоринговый балл выше порогового значения, система автоматически одобряет кредит.
   - Если балл ниже порога, дело передаётся кредитному специалисту для ручной оценки.

Результат:

- Если кредит одобрен, клиенту отправляется предложение с параметрами кредита (сумма, ставка, срок).
- Если кредит отклонён, клиенту отправляется уведомление с отказом.

5. **Оформление кредита**
   - Если клиент соглашается с предложением, он подписывает договор (электронно или в офисе).
   - Система создаёт кредитный счёт и отправляет уведомления в бухгалтерию и клиенту.
6. **Завершение процесса**
   - Клиент получает средства на указанный счёт.
   - Процесс завершается.

Задание

Построй BPMN-диаграмму для описанного процесса. Диаграмма должна содержать все ключевые шаги процесса. Обозначь возможные ошибки или исключения, например:

- Некорректные данные при заполнении формы.
- Недоступность внешнего сервиса (например, кредитного бюро).

Важно учитывать

- Полноту и корректность отображения всех этапов процесса.
- Логичность последовательности действий.
- Уместное использование элементов BPMN (пулы, задачи, гейтвеи, события).
- Учитывание исключений и альтернативных путей процесса.
- Читаемость (!) диаграммы.

Задание 3: Проектирование интеграций – Sequence diagram

Ты работаешь системным аналитиком в банке. Твоя задача — проанализировать процесс обработки платежа клиента и построить **Sequence Diagram**, отражающую взаимодействие микросервисов в этом процессе.

Сценарий задачи:

1. **Описание процесса:**
   - Клиент делает перевод через мобильное приложение банка.
   - Мобильное приложение отправляет запрос в микросервис обработки платежей (**Payment Service**).
   - **Payment Service**:

        - Проверяет баланс счета клиента через микросервис управления счетами (**Account Service**).
        - Если на счете клиента недостаточно средств, отправляет уведомление о недостаточном балансе.
        - Если средства есть, передает запрос в микросервис проверки лимитов и правил безопасности (**Fraud Detection Service**).
    - **Fraud Detection Service**:
        - Проверяет, не превышает ли сумма лимит для переводов и нет ли подозрений на мошенничество.
        - Возвращает результат проверки.
    - Если проверка успешна:
        - **Payment Service** уменьшает баланс отправителя через **Account Service**.
        - Увеличивает баланс получателя.
        - Отправляет подтверждение успешного перевода клиенту.
    - Если проверка неуспешна:
        - **Payment Service** отправляет уведомление клиенту о невозможности перевода (причина: превышен лимит или подозрение на мошенничество).

---

2. **Технические детали для реализации диаграммы:**
    - Выдели акторов процесса.
    - Покажи последовательность вызовов между микросервисами (учитывай, что взаимодействия могут быть по HTTP API, gRPC, Kafka).
    - Укажи ключевые данные, передаваемые между акторами (например, сумма перевода, статус проверки).
    - Отрази основные сценарии:
        1. Успешная обработка платежа.
        2. Недостаточно средств на счете.
        3. Платеж отклонен из-за превышения лимита или подозрения на мошенничество.
    - Учитывай, что **Fraud Detection Service** может работать асинхронно, а результат проверки поступает с небольшой задержкой.

---

3. **Формат сдачи:**
    - Построй Sequence Diagram с использованием UML-нотации.
    - Используй инструменты, такие как Lucidchart, Draw.io, PlantUML, или другие.

Дополнительное задание (опционально): Добавь в диаграмму обработку ошибок (например, недоступность одного из микросервисов) и продемонстрируй, как система должна обрабатывать такие ситуации.

Задание 3: Проектирование REST API

Твоя задача — спроектировать REST API для микросервиса обработки платежей (**Payment Service**) в системе, где клиент может осуществлять переводы между счетами.

Задача:

1. Спроектировать REST API для **Payment Service**, который выполняет следующие функции:
    - Создание нового перевода.
    - Получение статуса перевода.
    - Список всех переводов для указанного пользователя.
2. Определи:
    - **HTTP-методы и URL-эндпоинты** для каждой функции.
    - Формат запросов (JSON), включая обязательные и необязательные параметры. Контракт можно оформить в виде таблицы.
    - Формат ответов (JSON) для успешных и неуспешных операций, включая обработку ошибок. Контракт можно оформить в виде таблицы.
    - Приведи примеры запросов и ответов JSON к каждому эндопоинту.

    Если есть возможность, можно использовать инструменты Swagger/OpenAPI, для описания твоего API.

Задание 4: Проектирование БД

Необходимо спроектировать структуру реляционной базы данных для системы, обрабатывающей переводы между счетами клиентов. Эта база данных будет использоваться микросервисами, такими как **Payment Service** и **Account Service**, для выполнения и хранения информации о транзакциях.

Описание функционала системы:

- Клиенты банка имеют уникальные счета.
- Каждый перевод должен сохраняться в базе данных с информацией:
    - Отправитель.
    - Получатель.
    - Сумма.
    - Валюта.
    - Дата и время перевода.
    - Статус перевода (например, «успешен», «отклонен»).
    - Причина отказа (если перевод не выполнен).
- Баланс счета клиента должен обновляться при успешном переводе.
- Должна быть возможность отслеживать историю переводов для каждого клиента.

Требования к структуре базы данных:

1. Разработай схему базы данных, включающую:
    - Таблицу для хранения информации о клиентах банка.
    - Таблицу для хранения информации о счетах клиентов.
    - Таблицу для хранения данных о переводах.
    - Связи между таблицами, обеспечивающие целостность данных.

2. Продумай ключи и индексы:
    - Уникальные идентификаторы для клиентов, счетов и переводов.
    - Внешние ключи для связи между таблицами.
    - Индексацию для ускорения запросов (например, поиск переводов по клиенту или дате) – опционально.
3. Учет валюты:
    - Реализуй поддержку многовалютных переводов, добавив соответствующее поле в таблицу переводов.
    - Хранение курсов валют – опционально.

Задача:

1. Построй схему базы данных в виде ER-диаграммы, отражающую вышеуказанные требования.
2. Напиши SQL-запросы для:
    - Создания таблиц с учетом ключей, связей и индексов.
    - Добавления нового перевода в систему.
    - Обновления баланса счета после успешного перевода.
    - Получения списка переводов для указанного клиента за заданный период.

Формат сдачи:

- ER-диаграмма (можно использовать инструменты, такие как Lucidchart, Draw.io).
- SQL-скрипты для создания и управления базой данных.
- Примеры выполнения SQL-запросов.

Дополнительное задание (опционально): Добавь таблицу для хранения истории изменений баланса счета и реализуй механизм логирования таких изменений с помощью SQL-триггеров.

Задание 5: SQL запросы – easy level

Используй приложенную схему БД для выполнения заданий:

Таблица клиентов (clients)

id	name	email	phone	registration_date
1	John Doe	john@example.com	1234567890	2023-01-15
2	Jane Smith	jane@example.com	9876543210	2023-02-10
3	Bob Brown	bob@example.com	4567891230	2023-03-05

Таблица счетов (accounts)

id	client_id	account_number	currency	balance	status
1	1	111111	USD	5000.00	active
2	1	111112	EUR	3000.00	active
3	2	222222	USD	7000.00	active
4	3	333333	USD	0.00	active

Таблица переводов (transactions)

id	from_account_id	to_account_id	amount	currency	status	reason	created_at
1	1	3	500.00	USD	success	NULL	2023-05-01 12:00:00
2	2	3	300.00	EUR	failed	INSUFFICIENT_FUNDS	2023-05-02 15:30:00
3	3	2	200.00	USD	success	NULL	2023-06-01 10:15:00
4	1	4	1000.00	USD	success	NULL	2023-06-10 09:45:00

1. Вывести список всех клиентов, у которых есть активные счета в USD, с указанием их имени и текущего баланса.
2. Найти все переводы, которые завершились неудачей, и вывести номер счета отправителя, сумму, причину отказа и дату перевода.
3. Вывести общее количество переводов и их общую сумму в USD.
4. Вывести список всех клиентов, у которых на любом счете баланс меньше 1000, с указанием имени клиента и валюты счета.
5. Найти средний баланс для всех счетов в EUR.

Задание 6: SQL запросы – pro level

Используй приложенную схему БД для выполнения заданий:

Таблица клиентов (clients)

id	name	email	phone	registration_date
1	John Doe	john@example.com	1234567890	2023-01-15
2	Jane Smith	jane@example.com	9876543210	2023-02-10
3	Bob Brown	bob@example.com	4567891230	2023-03-05

Таблица счетов (accounts)

id	client_id	account_number	currency	balance	status	last_update
1	1	111111	USD	5000.00	active	2023-06-10 09:00:00

id	client_id	account_number	currency	balance	status	last_update
2	1	111112	EUR	3000.00	active	2023-06-11 14:00:00
3	2	222222	USD	7000.00	active	2023-06-10 10:00:00
4	3	333333	USD	0.00	active	2023-06-12 08:00:00

Таблица переводов (transactions)

id	from_account_id	to_account_id	amount	currency	status	reason	created_at
1	1	3	500.00	USD	success	NULL	2023-05-01 12:00:00
2	2	3	300.00	EUR	failed	INSUFFICIENT_FUNDS	2023-05-02 15:30:00
3	3	2	200.00	USD	success	NULL	2023-06-01 10:15:00
4	1	4	1000.00	USD	success	NULL	2023-06-10 09:45:00

Таблица истории баланса (balance_history)

id	account_id	change_amount	change_type	change_date
1	1	-500.00	withdrawal	2023-05-01 12:00:00
2	2	-300.00	withdrawal	2023-05-02 15:30:00
3	3	+200.00	deposit	2023-06-01 10:15:00
4	1	-1000.00	withdrawal	2023-06-10 09:45:00
5	4	+1000.00	deposit	2023-06-10 09:45:00

1. Найти текущий баланс каждого счета, посчитав все изменения из таблицы balance_history с использованием оконной функции.
2. Найти 3 последних успешных перевода (по времени), используя оконные функции.
3. Создать триггер для обновления баланса в таблице accounts после добавления записи в таблицу balance_history.
4. Вывести топ-2 клиентов с наибольшей общей суммой переводов (учитывая и исходящие, и входящие переводы).
5. Создать хранимую процедуру, которая для указанного клиента возвращает список всех его счетов и их текущий баланс.
6. Найти разницу в балансе каждого счета между последним и предпоследним изменением.