Предметная область: Волшебный банк Гринготтс

Описание: **GWBMS** - это специализированная банковская система, предназначенная для обслуживания клиентов магического мира. Система обеспечивает хранение и управление финансовыми средствами клиентов, а также предоставляет услуги по хранению и учету магических артефактов.

Проблема: В магическом мире вселенной Гарри Поттера существует один из крупнейших банков — банк Гринготтс, в котором работает огромная куча банкиров, преимущественно руками. Их задача это отслеживание хранимых магических артефактов, управление счетами(как денежными средствами), учет и проведение различных транзакций, управление персоналом, создание отчетностей, а также им необходимо хранить информацию об артефактах, которые они хранят(их историю и прочую мелочь)

Решение:

Для решения этих задач предлагаем разработать **Gringotts Wizarding Bank Management System (GWBMS)** — современную автоматизированную систему для магического банка Гринготтс, которая позволит сотрудникам банка:

1. Управление счетами и транзакциями (в частности денежными и возможность положить в банковскую ячейку артефакт)
2. Валидация возможности хранения того или иного артефакта(возможен отказ в случае,если артефакт несет определенную ценность в мире магии
3. Выдача ключей к банковским ячейкам и отслеживание, кому эти ключи были выданы
4. Возможность создавать отчетность о прошедших денежных транзакциях и текущих артефактов, с дальнейшей возможностью экспорта(например, в xlsx формат)

Требования

Функциональные требования

1. **Управление счетами и транзакциями**
    - Система должна поддерживать различные магические валюты

        Типы валют:

        1. Галеоны
        2. Еслеоны
        3. Рулеоны
    - Система должна предоставлять возможность клиентам открывать и управлять мультивалютными счетами, включая магические валюты. Понятие управление счетом включает в себя:
        - Создание нового счета
        - Внесение валют на новый счет
        - Снятие валют с счета
        - Перевод средств на другой счет
            - Счет другого клиента
            - Свой другой счет
    - Система должна поддерживать процесс хранения артефактов в банковских ячейках, включая создание записи об артефакте, описание и регистрационные данные.
    - Система должна разрешать или запрещать хранение артефактов в зависимости от их ценности и уровня опасности (с поддержкой магических критериев валидации).

        Магические критерии валидации включают в себя

        1. Предыдущий владелец не был осужден за особо тяжкое преступление
        2. Уровень опасности, присвоенный министерством магии, не превышает норму
2. **Управление артефактами**
    - Система должна хранить информацию о каждом артефакте

        Каждый артефакт должен хранить в себе следующую информацию:

        1. история
        2. текущий владелец
        3. магические свойства
        4. уровень опасности(последняя запись, выданная министерством магии)
    - Система должна автоматически проводить проверку валидности артефакта для хранения в ячейке (с учетом законов магического мира).
        i. Валидироваться система должна при помощи prolog
    - Система должна предоставлять возможность сотрудникам создавать и обновлять информацию об артефактах в банковской системе.

Обновление информации подразумевает

1. Изменение текущего владельца
2. Добавление/удаление магических свойств
3. Обновление уровня опасности(запрос на обновление приходит на министерство магии)

3. **Выдача ключей к банковским ячейкам**
    - Система должна генерировать и отслеживать магические ключи, выданные для доступа к ячейкам.

      Ключ обладает следующими характеристиками

      1. Он уникален
      2. Выдается на сутки(после чего перестает быть валидным)
      3. Задается в виде JWT токена,
    - Система должна отслеживать историю выдачи ключей и лиц, имеющих к ним доступ.
4. **Отчетность и экспорт данных**
    - Система должна генерировать отчеты по финансовым транзакциям.

      Отчет должен содержать

      1. историю операций
      2. баланс счета
      3. тип валюты
    - Система должна генерировать отчеты по артефактам, хранящимся в банке, с возможностью фильтрации по статусу, владельцу и магическим свойствам.
    - Отчеты должны поддерживать экспорт в форматы XLSX, PDF и csv

Нефункциональные требования

1. **Производительность**
    - Система должна обеспечивать обработку запросов от пользователей за время не более 5 секунд.
    - Система должна поддерживать одновременную работу не менее 100 пользователей без потери производительности.
2. **Масштабируемость**
    - Система должна легко масштабироваться с ростом количества клиентов и транзакций.
3. **Безопасность**
    - Доступ к информации о клиентах, их счетах и артефактах должен быть защищен с использованием двуфакторной аутентификации. (с помощью чего - вернитесь с одобрением)
    - Система должна поддерживать вход через протокол Oauth2

    - Система должна поддерживать разграничение прав доступа на основе ролей сотрудников (гоблинов), с возможностью управления этими ролями.
    - Все данные должны быть зашифрованы, как при хранении, так и при передаче.
4. **Надежность**
    - Система будет использовать бэкапы для возможности откатить базу данных в случае неполадок
    - Для мониторинга будет использована Grafana
5. **Юзабилити**
    - Интерфейс системы должен быть интуитивно понятен пользователям, с четкой навигацией и инструкциями

      Под четким и интуитивно понятным интерфейсом принимается такой интерфейс, который придерживается всех эвристик Нильсона

    - Система должна поддерживать многоязычие(англ+русский)
6. **Совместимость**
    - Экспортированные отчеты должны корректно открываться в популярных офисных пакетах (ОТКРЫТИЕ В МОЙ ОФИС БЕЗ ПРОБЛЕМ)

Описание прецедентов

Прецедент: Операции с магическими артефактами

ID: 1

Краткое описание: Хранение, проверка и управление магическими артефактами в банковских ячейках.

Главный актор: Клиент и Сотрудник банка

Предусловия: Пользователь авторизован в системе.

Основной поток:

1. Клиент приходит в банк и хочет выбрать услугу "Добавить артефакт в ячейку".
2. Сотрудник банка вводит данные артефакта и отправляет данные об артефакте на проверку.
3. Сотрудник банка просматривает результат проверки.
4. Если артефакт одобрен, он помещается в ячейку.
5. Клиент может изъять артефакт из ячейки при необходимости.

Альтернативный поток:

- Если артефакт не соответствует требованиям, система уведомляет клиента об отказе.
- **Постусловия:** Артефакт хранится в ячейке или был изъят клиентом.

Прецедент: Создание и экспорт отчетов

ID: 2

Краткое описание: Генерация отчетов о финансовой деятельности и состоянии артефактов.

Главный актор: Сотрудник банка / Администратор

Предусловия: Пользователь авторизован в системе.

Основной поток:

1. Пользователь запрашивает опцию "Создать отчет".
2. Система предоставляет выбор параметров для отчетов (по клиентам, артефактам и т.д.).
3. Пользователь задает параметры и генерирует отчет.
4. Пользователь может экспортировать отчет в формат XLSX или PDF.
   Альтернативный поток: -
5. **Постусловия:** Пользователь получает отчет и может его сохранить.

Прецедент: Управление доступом к ячейкам

ID: 3

Краткое описание: Выдача и отслеживание ключей к банковским ячейкам.

Главный актор: Сотрудник банка

Предусловия: Пользователь авторизован в системе.

Основной поток:

1. Сотрудник выбирает опцию "Сгенерировать ключ к ячейке".
2. Система запрашивает информацию о клиенте и ячейке.
3. Сотрудник выдает цифровой ключ клиенту и регистрирует информацию в системе.
4. Сотрудник может отслеживать выданный статус ключа через систему.
   Альтернативный поток: Пользователь не имеет прав на получения ключа и следует отказ в этой операции

Постусловия: Клиент получает ключ, и информация об этом фиксируется в системе.

ID: 4

Краткое описание: Открытие, управление и выполнение операций с банковскими счетами.

Главный актор: Сотрудник банка

Предусловия: Пользователь авторизован в системе.

Основной поток:

1. Сотрудник банка получает данные от клиента
2. Сотрудник вводит их в систему, которая валидирует значения
3. Сотрудник выполняет операцию, запрошенную пользователем

Альтернативный поток:

Если данные неверны, система уведомляет о необходимости исправления информации.

Планируемые затраты по часам

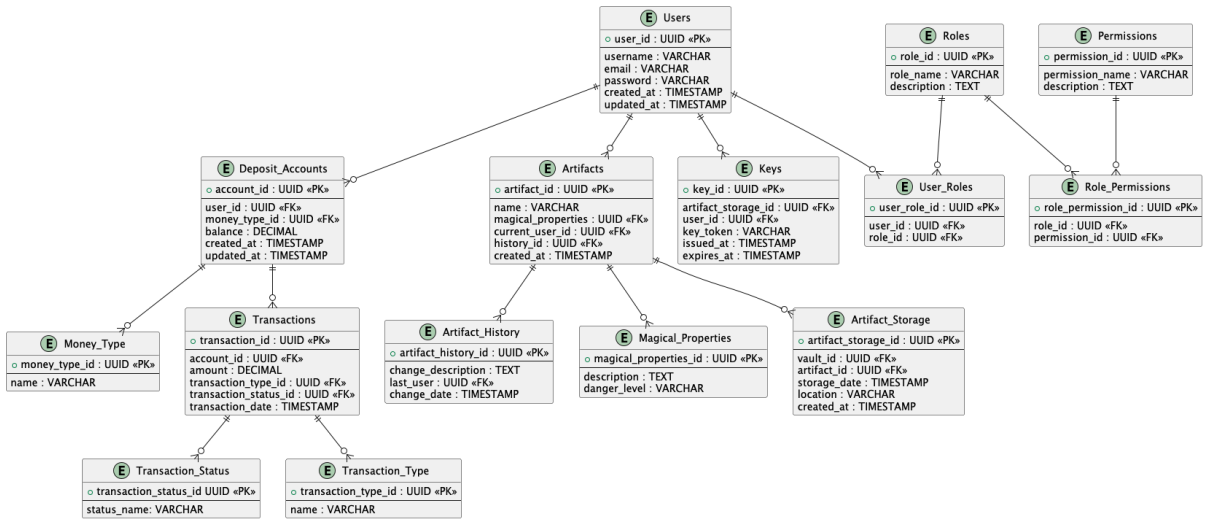
Требование	Оценка времени (часы)
Управление счетами и транзакциями	
Поддержка различных магических валют	8
Открытие и управление мультивалютными счетами	12
Создание нового счета	6
Внесение валют на новый счет	6
Снятие валют с счета	6
Создание депозита	8
Перевод средств	10
Хранение артефактов	
Поддержка хранения артефактов	10
Проверка ценности и уровня опасности	8

Управление артефактами	
Хранение информации об артефактах	8
Автоматическая проверка валидности	10
Обновление информации об артефактах	8
Выдача ключей к ячейкам	16
Отчетность и экспорт данных	
Генерация отчетов по транзакциям	6
Генерация отчетов по артефактам	6
Поддержка экспорта отчетов	10

—

Второй этап

UML диаграмма:




```sql
-- 1. Поиск артефактов в хранилище:

CREATE INDEX idx_artifact_in_storage ON Artifact_Storage
(vault_id, storage_date);

-- Преимущества: Ускоряет выборку артефактов по ID, местоположению
и дате хранения.



--  2. Проверка артефакта на валидность магических свойств:

CREATE INDEX idx_magical_properties ON Magical_Properties
(danger_level);

-- Преимущества: Оптимизирует поиск артефактов с определёнными
уровнями опасности.




-- Предлагаемые функции

-- 1) Генерация отчета по счетам

-- 2) Операция средств между счетами
```

https://github.com/JABAN111/SecondPart/blob/main/init_scripts/script.sql

—

первый запрос без использования индексации 1

psql=# EXPLAIN ANALYZE

SELECT *

FROM Artifact_Storage

WHERE artifact_id = '42da586e-2da1-43de-b219-a9d8e669e63a'

  AND vault_id = '53c9a9b5-8fd6-40c1-8c25-0b38741e8021'

  AND storage_date = '2024-11-18';

QUERY PLAN

Seq Scan on artifact_storage (cost=0.00..279.16 rows=1 width=72) (actual time=3.599..3.599 rows=0 loops=1)

  Filter: ((artifact_id = '42da586e-2da1-43de-b219-a9d8e669e63a'::uuid) AND (vault_id = '53c9a9b5-8fd6-40c1-8c25-0b38741e8021'::uuid) AND (storage_date = '2024-11-18 00:00:00'::timestamp with

out time zone))

  Rows Removed by Filter: 10009

Planning Time: 0.638 ms

Execution Time: 3.698 ms

(5 rows)

С использованием

QUERY PLAN

Index Scan using idx_artifact_in_storage on artifact_storage (cost=0.29..8.31 rows=1 width=72) (actual time=0.105..0.105 rows=0 loops=1)

Index Cond: ((artifact_id = '42da586e-2da1-43de-b219-a9d8e669e63a'::uuid) AND (vault_id = '53c9a9b5-8fd6-40c1-8c25-0b38741e8021'::uuid) AND (storage_date = '2024-11-18 00:00:00'::timestamp

without time zone))

Planning Time: 1.177 ms

Execution Time: 0.287 ms

(4 rows)

До создания индекса

EXPLAIN ANALYZE

SELECT *

FROM Magical_Properties

WHERE danger_level = 'High';

QUERY PLAN

Seq Scan on magical_properties (cost=0.00..229.11 rows=10003 width=55) (actual time=0.056..5.194 rows=10003 loops=1)

Filter: ((danger_level)::text = 'High'::text)

Rows Removed by Filter: 6

Planning Time: 1.413 ms

Execution Time: 5.934 ms

(5 rows)

После создания индекса

QUERY PLAN

Seq Scan on magical_properties (cost=0.00..229.11 rows=10003 width=55) (actual time=0.013..2.402 rows=10003 loops=1)

  Filter: ((danger_level)::text = 'High'::text)

  Rows Removed by Filter: 6

Planning Time: 0.684 ms

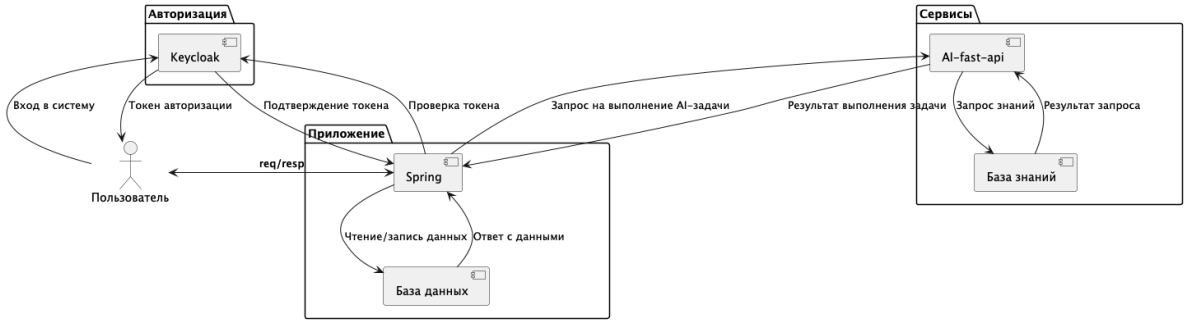
Execution Time: 2.903 ms

(5 rows)

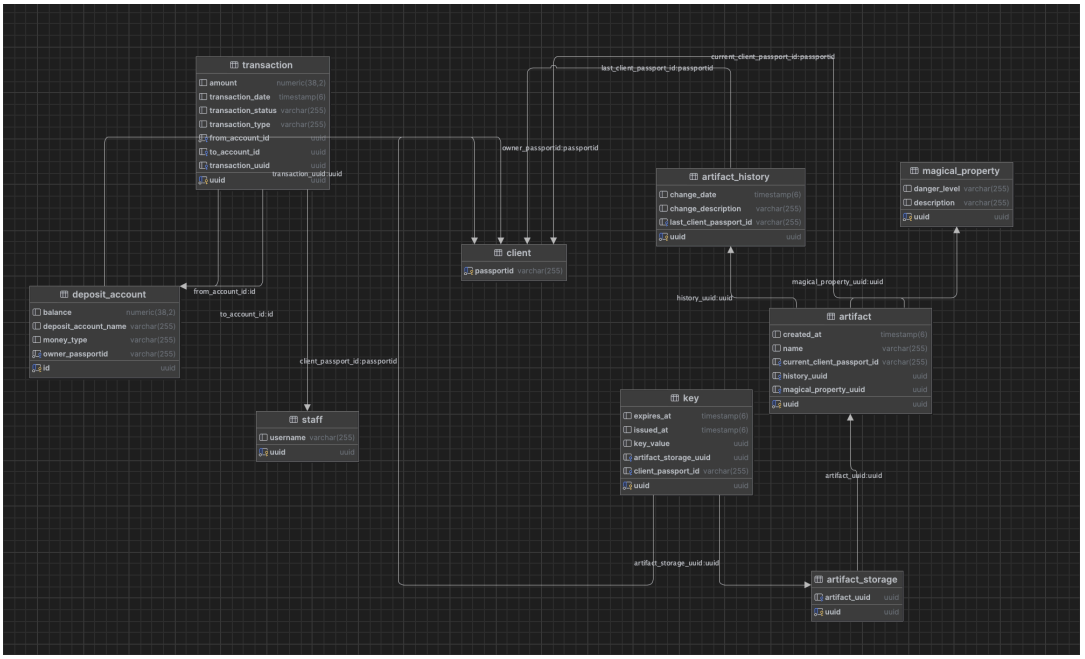
—

Третий этап:

Изобразить диаграмму классов, представляющую общую архитектуру системы.



Реализовать уровень хранения информационной системы на основе разработанной на предыдущем этапе базы данных.



При реализации уровня хранения должны использоваться функции/процедуры, созданные на втором этапе с помощью pl/pgsql. Нельзя замещать их использование альтернативной реализацией аналогичных запросов на уровне хранения информационной системы.

Функция:

```
create or replace function
get_account_transactions_multiple_types(acc_id uuid,
transaction_types text[])

   returns TABLE(moneyType text, accountId uuid,
transactionAmount numeric, transactionType text)

   language plpgsql

as

$$

BEGIN

   RETURN QUERY

       SELECT
```

```
            ac.money_type::TEXT AS moneyType,
            tr.to_account_id AS accountId,
            tr.amount AS transactionAmount,
            tr.transaction_type::TEXT AS transactionType
        FROM
            deposit_account ac
                LEFT JOIN
            transaction tr
            ON
                ac.id = tr.from_account_id
        WHERE
            ac.id = acc_id
          AND tr.transaction_type = ANY(transaction_types);
END;
$$;
CREATE OR REPLACE FUNCTION transfer_funds(
   from_account_id UUID,
   to_account_id UUID,
   amount NUMERIC(38,2)
) RETURNS VOID AS $$
BEGIN
   IF amount <= 0 THEN
        RAISE EXCEPTION 'Transfer amount must be positive';
   END IF;
```

```
   IF (SELECT balance FROM deposit_account WHERE id =
from_account_id) < amount THEN

       RAISE EXCEPTION 'Insufficient funds in the source
account';

   END IF;



   UPDATE deposit_account

   SET balance = balance - amount

   WHERE id = from_account_id;



   UPDATE deposit_account

   SET balance = balance + amount

   WHERE id = to_account_id;



END;

$$ LANGUAGE plpgsql;
```