Фінальний проект з курсу МООП - ATM

Виконала команда "Я вбив козу кулаком"

- Тихий Роман
- Ярошепта Богдан

Розподілення обов'язків

- Тихий Роман - створення бекенду, робота з базою, робота з прототипом
- Ярошепта Богдан - створення фронтенду, робота з прототипом

Опис проекту

Цей документ краще переглядати на гітхабі у форматі .md.

Проект полягав у створенні віртуального банкомату.

Подальші терміни використані у описі проекту:

- ATM - банкомат. Виконує роль імітатора справжнього банкомату. Пов'язаний з банком. Many-to-many relationship. Один банкомат може обслуговувати картки багатьох банків.
- Bank - банк. Банк відповідальний за зв'язок із банкоматами(Many-to-many) та картками(One-to-many). Один банк може мати багато банкоматів та багато карток.
- User - користувач, клієнт банку. Кожен банк має свого унікального користувача, тому зв'язок між клієнтом і банком - Many-to-one, користувач може бути клієнтом лише одного банку.
- Card - картка. Картка використовується для аутентифікації у банкоматі, а також для верифікації банківських операцій. Картка є холдером рахунків. Зв'язок картки із користувачем - Many-to-one. Зв'язок із користувачем та карткою - One-to-many: один користувач може мати багато рахунків, та до картки може бути прив'язано багато рахунків.
- Account - рахунок. Є центральним та найбільш важливим об'єктом у архітектурі проєкт. Рахунок може бути Transactional(в різних регіонах світу має різну назву, але загальна назва - Transactional account: транзакційний, поточний(current), розрахунковий(checking)), Saving(зберігальний, розрахунковий).
    - Транзакційний рахунок може бути кредитним, або ні. Кредитний має ліміт, при перевищенні якого буде нарахована комісія в залежності від обраного плану. Також кожна картка завжди має транзакційний рахунок за замовчуванням(default). Рахунок за замовчуванням використовується для операцій зняття готівки в банкоматі, а також поповнення.
    - Зберігальний рахунок використовується для накопичення коштів. На відміну від транзакційного не має кредитного ліміту, а також має певні додаткові обмеження: поповнення зберігального рахунку можливе лише з іншого рахунку власника зберігального рахунку та лише за умови, що план накопичення рахунку передбачає можливість додаткового поповнення під час накопичення або рахунок в даний момент часу не здійснює накопичення за жодним із планів. Переказувати з рахунку можна на будь-які інші рахунку, якщо умови отримувача підходять. Якщо накопичувальний план не активований, є можливість обрати інший план, або почати накопичення з обраним. Під час накопичення мануальні перекази з рахунку заборонені. Передбачена можливість капіталізації та нарахування процентів на картку. Капіталізація - проценти після виплати(період виплати передбачений накопичувальним планом) додаються до накопичуваної суми, збільшуючи при цьому фінальний дохід. Альтернатива до капіталізації - переказ процентів на розрахункову картку, фінальний дохід не зросте. Капіталізація увімкнена за замовчуванням. Також передбачена можливість автоматичного продовження плану. Після закінчення повного періоду, рахунок знову почне накопичення за попередніми умовами. Період розрахунку проценту - 1 день.
- Transaction - транзакція або ж переказ. Друга за важливістю сутність у проєкт. Відповідає за відображення банківських операцій здійснених користувачем. Транзакція може бути однією з типів: переказувальна(Transferring), знімальна(Withdraw), поповнюючи(Deposit), регулярна або відкладена(Regular). Зв'язок із рахунками - Many-to-one.
    - Transferring - використовується для переказування грошей з одного рахунку на інший. Може мати комісію, якщо переказувач перевищий свій кредитний ліміт. При знятті готівки в банкоматі комісія нараховується також.
    - Scheduled - транзакція, для якої можна обрати час виконання. Під час створення транзакції, перевіряються всі умови транзакції, а також при її безпосередньому виконанні: якщо наприклад при створенні транзакції на рахунку відправника було достатньо грошей, а при виконанні вже ні, то транзакція не виконається зі статусом REJECTED.
    - Regular - транзакція, для якої можна обрати частоту виконання, кількість повторів, та опціонально час початку першої транзакції, як і для відкладеної транзакції має перевірку як на створенні та і при

виконанні кожного повтору. Може мати необмежену кількість повторень.
- LoginAttempt - відповідає за збереження історії аутентифікації користувача у банкоматі. Якщо протягом останньої години користувач ввів пін-код тричі поспіль неправильно - картка буде заблокована. Також зберігає історія про те в якому саме банкоматі була здійснена аутентифікація.

Діаграми класів та таблиць бази даних

Діаграми були створені у різних форматах за допомогою IntelliJ IDEA. Їх можна знайти тут.

Функціонал

Список можливостей доступних користувачу

- Обрати банкомат, що обслуговує картку його банку
- Здійснити вхід у банкомат за допомогою номеру картки та пін-коду
- Здійснити вихід із сесії банкомату
- Переглянути інформацію про картку, себе та поточну сесію
- Змінити пін-код
- Зняти гроші із рахунку за замовчуванням
- Покласти гроші на рахунок за замовчуванням
- Створити нову переказувальну транзакцію. Конвертація валют та нарахування комісії за перевищення ліміту здійснюються автоматично. Конвертація не бере додаткової комісії.
- Створити нову відкладену транзакцію
- Створити нову регулярну транзакцію із відкладеним стартом чи ні
- Переглянути всі свої рахунки
- Переглянути інформацію про конкретний рахунок
- Переглянути історію платежів з усіх рахунків
- Переглянути історію платежів з конкретного рахунку
- Переглянути відкладені та регулярні платежі створені з конкретного рахунку
- Відмінити регулярні та відкладені платежі для конкретного рахунку
- Створити рахунок із планом обраним планом або без плану(Plain). Для рахунку можна обрати назву та валюту. Після створення рахунку валюту змінити не можна. Обмеження на кількість рахунку для картки не існує.
    - Транзакційний
    - Накопичувальний
- Змінити план накопичувального рахунку
- Розпочати накопичення за обраним планом
- Розірвати контракт та припинити накопичення(при цьому користувач не отримає зароблені вже проценти не важливо чи була капіталізація чи ні, якщо були обрані проценти на картку, сума всіх транзакцій здійснених з накопичувального рахунку у період зі старту накопичення та до розірвання контракту буде вирахувана із суми вкладу)

Створення прототипу та документації

Для вдалої комунікації між розробкою бекенду та фронтенду ми створили прототип за допомогою Figma(копія доступна тут). Прототип містить фрейми, які є інтерактивними, та за допомогою яких можна представити функції компонентів(наприклад кнопок). Також робоча дошка містить коментарі-підказки, посилання на HTTP ендпоінти, документи з REST документацією. Дизайну прототипу не схожий на фінальний результат, адже його суть не в цьому.

REST документація генерувалась за допомогою таких бібліотек

```
    <dependency>
        <groupId>org.springframework.restdocs</groupId>
        <artifactId>spring-restdocs-mockmvc</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>capital.scalable</groupId>
        <artifactId>spring-auto-restdocs-core</artifactId>
        <version>2.0.11</version>
        <scope>test</scope>
    </dependency>
```

А також після автогенерації редагувалась вручну за потреби

Тести

Бекенд має 29 інтеграційних тестів які покривають функціонал проекту. Тести розроблялись за допомогою фреймворків JUnit та spring-mockmvc. Також REST документація генерувалась за допомогою цих тестів, тому так би мовити двох зайців одним пострілом.

Для тестів використовується окремий spring.active.profile=test, цей профіль переписує стандартну конфігурацію, а саме база змінюється з PostgresQl на H2, яка працює у in-memory режимі, тобто дані крихкі, та після перезапуску застосунку не зберігаються. База повністю написана на Java без використання бібліотек тому легко може бути вбудованою в JVM process, а сам розмір бази у jar форматі складає ~2.5mb. Для заповнення тестовими даними був написаний клас DbDataInserter

Графічний інтерфейс

Графічний інтерфейс був створений у стилі Windows98 з використання js/css/html бібліотек. Користувацький інтерфейс містить валідацію вхідних даних користувача, а також обробку помилок від сервера.

Технологічний стек

- Середовища розробки
  - IntelliJ IDEA - бекенд, роботу з базою, діаграми, написання цього markdown файлу
  - VSCode - фронтенд
- Бази даних - ази виключно реляційні, адже предметна область вимагає контролю та безпеки, а NoSQL бази не здатні забезпечити ACID.
  - PostgresQl - для розробки
  - H2 - для тестів
- Фреймворки
  - Spring Boot - гарне рішення для для розробки HTTP сервера та не тільки
  - AngularJs - перевірений часом JS фреймворк
  - JUnit - одне з найкращних рішень для юніт тестів мовою програмування Java