проект на тема:

Система за обработка и управление на клиентска бановка информация

колектив:
F74277 - Мартин Драганов
F76731 - Илиян Костов
F78397 - Николай Николов

част: сървър - база данни

разработена от Николай Николов (F78397)
https://github.com/niksanElements/Bank-System

Проектът включва:

- Създаване на графичен потребителски интерфейс за клиентите на банката, генериращ заявки и интерпретиращ отговорите - разработен от Мартин Драганов (F74277)
- Осигуряване на мрежова фукнционалност, управление на сървъра и защита на преноса на данни - разработена от Илиян Костов (F76731)
- Предоставяне на връзка от сървъра към система за управление на база данни (MySQL) за съхранение и обработка на транзакциите - разработена от Николай Николов (F78397)

Разработената от студента част включва:

- Анализ на бизнес логиката на системата за обработка и управление на клиентска банкова информация.
- Анализ и създаване на релационна схема на базата данни (съгласно бизнес логиката) - определяне на структурата на съхраняваните данни от СУБД (система за управление на базата данни).
- Създаване на процедура за обработка на заявките от клиентите съгласно мрежовия протокол.
- Създаване и управление на връзка от сървъра към СУБД.

Съдържание:

Списък на изображенията:

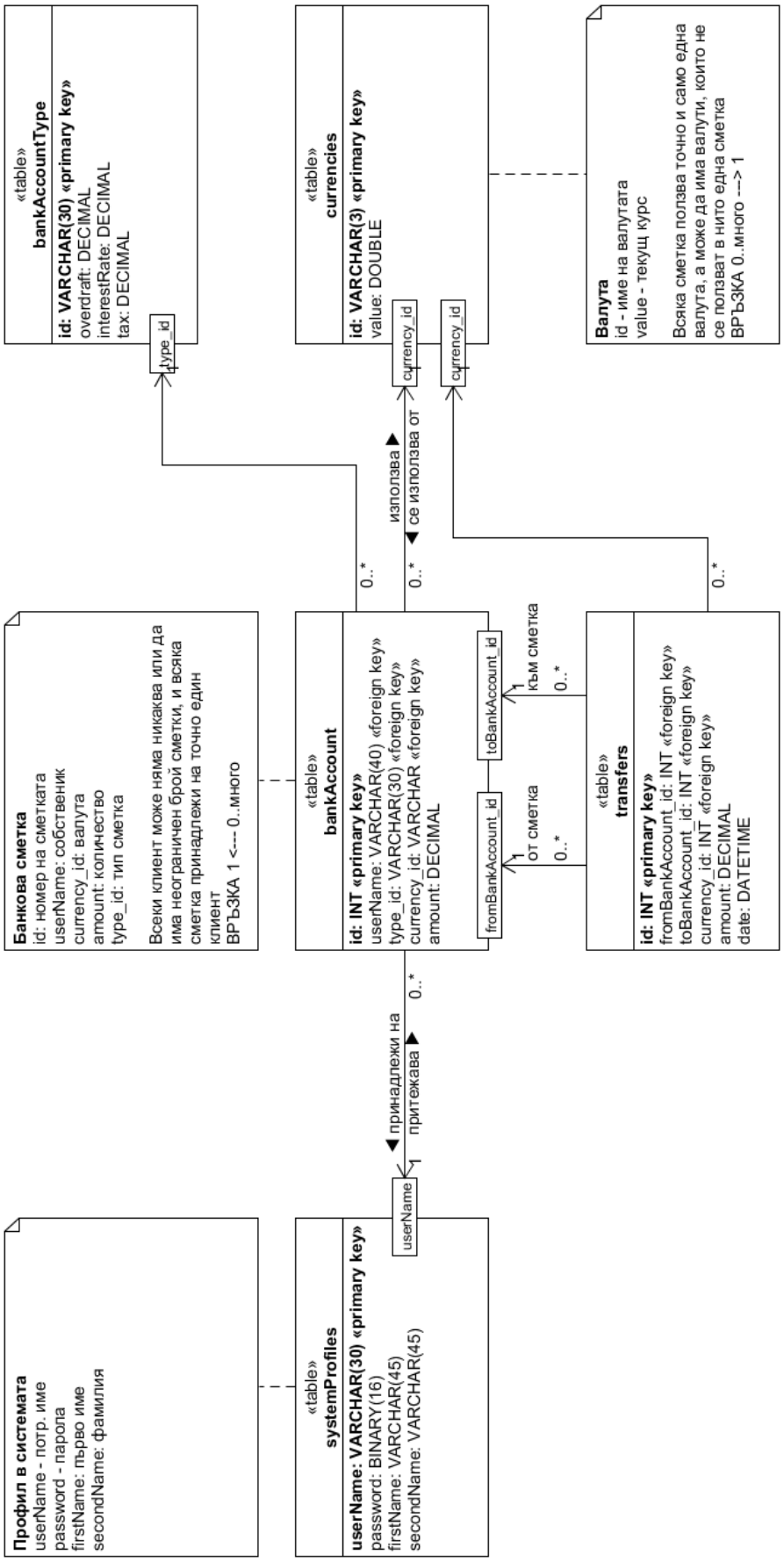
1. Анализ на бизнес логиката

- банковата система обслужва клиенти, които използват системата, идентифицирайки се уникално чрез потребителски профил
- всеки потребителски профил може да управлява неограничен брой банкови сметки
- всяка банкова сметка има определен вид (спестовна, разплащателна,кредит и др.) и използва определена валута
- всяка валута има текуща относителна стойност (обменни курсове)
- между банковите сметки могат да се извършват банкови транзакции - в банковите сметки може да се внасят или теглят пари в зависимост от вида

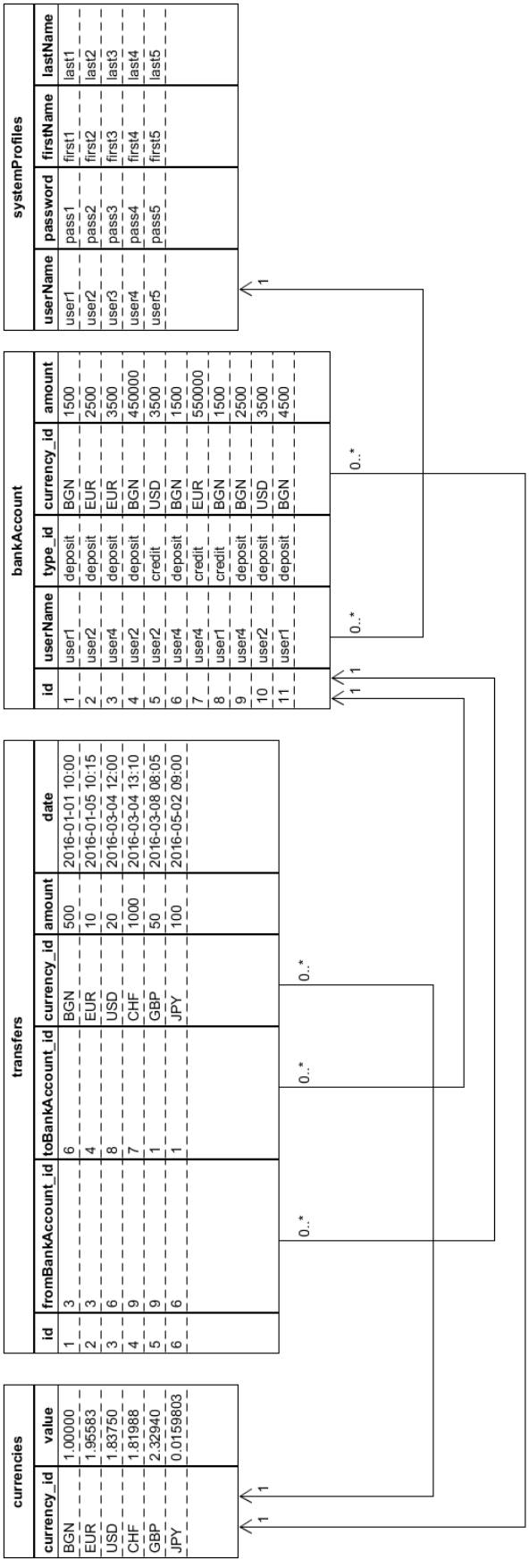
2. Релационна схема на базата

Обособяването на пет вида обекти предполага използването на база данни с модел, основан на пет релации (таблици):

- потребителски профили:
  - потребителско име (главен ключ)
  - парола
  - първо име
  - фамилия
- банкови сметки:
  - ID (главен ключ)
  - потребителско име (чужд ключ)
  - вид на сметката (чужд ключ)
  - валута (чужд ключ)
  - наличност
- вид на сметката:
  - вид (главен ключ)
  - лихвен процент
- валути:
  - вид валута (главен ключ)
  - текуща относителна стойност - съгласно курсовете за обмен
- банкови транзакции:
  - ID (главен ключ)
  - сметка-източник (чужд ключ)
  - сметка-направление (чужд ключ)
  - валута (чужд ключ)
  - количество
  - дата



Фиг. 1 Релационна схема на базата данни



Фиг. 2 Примерно състояние на базата данни

3. Процедура за обработка на клиентски заявки - псевдокод:

```
при получено съобщение "message":

ако (тип съобщение = LOGINREQUEST)
{
        SELECT * FROM Потребители
        WHERE потребителскоИме = message.getLoginUsername()
        AND парола = message.getLoginPassword();
        ако има - успешен логин и данни, иначе - неуспешен логин
}
иначе ако (тип съобщение = REGISTERREQUEST)
{
        INSERT
        INTO Потребители (потребителскоИме, парола, име, фамилия)
        VALUES (    message.getRegisterUsername(),
                    message.getLoginPassword(),
                    message.getFirstName(),
                    message.getLastName()          );
        ако успех - успешен логин и данни, иначе - неуспешен логин;
}
иначе
{
        обработка наредена от клиента операция;
        връщане на обновените данни за клиента (виж раздел 4);
}
```

4. Връщане на обновените данни за клиента - псевдокод:

1. създаване на капсулиращ данните обект:
   data = new ProfileData();
2. въвеждане наличности по банкови сметки:
   SELECT * FROM БанковиСметки
   WHERE Потребители.потребителскоИме = message.getUsername();
   в цикъл по редовете от резултата: {
      data.addToBalance(// добавяне на ред за сметката
         "ID сметка",
         "тип сметка",
         "валута",
         "количество");
   }
3. въвеждане история на транзакциите:
   SELECT * FROM История
   WHERE Потребители.потребителскоИме = message.getUsername();
   в цикъл по редовете от резултата: {
      data.addToTransferHistory(// добавяне на ред за история
         "ID транзакция",
         "от сметка",
         "до сметка",
         "валута",
         "количество",
         "дата");
   }
4. въвеждане валутни курсове:
   създаване converter = new CurrencyConverter();
   SELECT * FROM ВалутниКурсове;
   в цикъл по редовете от резултата: {
      converter.set("валута", "стойност"); // задаване курс
   }
   data.currencyConverter = converter;
5. връщане резултат:
   return data;

5. Създаване и управление на връзка от сървъра към СУБД

Създава се обект, имплементиращ интерфейса MessageHandler (описан в частта за мрежовия протокол). Този обект (интерфейс) дефинира специфична обработка за всеки вид входящо съобщение. След като бъде получено по мрежата, входящото клиентско съобщение-заявка се разпознава като вид по полето си message.type. В зависимост от типа съобщение автоматично се изпълнява дефинираната обработка и се връща резултат - обновените данни за клиента, при условие, че клиентът вече е успешно вписан в системата (достъпът се контролира от сървъра).

Така създаденият обект се предава на конструктора на сървъра, като по този начин се указва на сървъра да използва така дефинираните обработки за всички входящи съобщения от клиенти.

Обработките на клиентските съобщения (заявки) се извършват в метода за обработка като синхронизирани (synchronized). По тои начин едновременно се гарантира както безопасност от допускане на некоректни заявки към базата, така и обработка на заявките в реда на постъпването им.