МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра Прикладная математика и информатика
(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(наименование профиля, специализации)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка ПО для автоматизации денежных переводов поставщикам услуг»

Обучающийся С. А. Бойчакаев _____
(И.О. Фамилия) (личная подпись)

Руководитель С.В. Митин
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент, А.В. Егорова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2023

Аннотация

Тема бакалаврской работы: «Разработка ПО для автоматизации денежных переводов поставщикам услуг».

Цель работы – разработка программного обеспечения автоматизации денежных переводов поставщикам услуг.

Объект – система оплаты услуг ЖКХ.

Предмет – программное обеспечение автоматизации денежных переводов поставщикам услуг ЖКХ.

Выпускная квалификационная работа состоит из введения, 3-х глав, заключения, списка литературы, состоящего из 20 источников. Работа содержит 43 страницы 10 таблиц и 14 рисунков.

Во введении обоснована актуальность темы, установлены цель, задачи, объект и предмет работы.

В первой главе выполнены постановка задачи на разработку ПО для автоматизации денежных переводов поставщикам услуг, формирование требований к ПО и сравнительный анализ используемых аналогов.

Во второй главе выполнено проектирование ПО для автоматизации денежных переводов поставщикам услуг, в ходе которого проведен выбор архитектуры ПО, разработаны логической модели ПО и алгоритм формирования перевода из платежей.

В третьей главе проведены выбор средств разработки, реализация и тестирование приложения.

В заключении представлены краткие результаты работы и дана оценка выполнения задач и достижения цели.

Abstract

The title of the bachelor's thesis is: "Developing software to automate money remittance to service providers".

The purpose of the work is to develop software to automate money remittance to service providers.

The object is the system of payment for housing and communal services.

Subject - software automation of money remittance to providers of housing and communal services.

Graduation qualification work consists of an introduction, 3 chapters, a conclusion, a list of references, consisting of 20 sources. The work has 43 pages 10 tables and 14 figures.

In the introduction justified relevance of the theme, set the goal, objectives, object and subject matter.

The first chapter contains statement of task for development of software for automation of money remittance to service providers, formation of requirements to software and comparative analysis of used analogues.

The second chapter deals with designing software for the automation of money remittances for service providers, in the course of which software architecture was selected, a logical software model was developed and an algorithm for forming a remittance from payments was developed.

The third chapter includes selection of development tools, implementation and testing of the application.

In the conclusion, a summary of the results of the work is presented and an assessment of the fulfilment of the tasks and achievement of the objective is given.

Содержание

Введение

В настоящее время внедрение информационных технологий в сферу платежей ЖКХ является важным и востребованным направлением. Информатизация сферы ЖКХ в целом позволяет создать качественно новую систему управления ЖКХ, а также систему регламентированного взаимодействия с исполнительными органами государственной власти, преодолеть разрыв между уже относительно развитой нормативной базой и правоприменительной практикой, а также повысить качество принимаемых решений, социальную защищенность населения и усилить контроль за жилищно-коммунальной сферой деятельности.

Использование информационных технологий в сфере ЖКХ направлено на решение следующих задач:

– повышение оперативности диспетчеризации;

– обработка информации о техническом состоянии жилого фонда территории;

- дистанционное управление объектами ЖКХ;

- моделирование ситуаций;

- бухгалтерский учет и расчет оплаты за коммунальные услуги;

- повышение качества работы с населением;

- информационное обслуживание органов муниципального управления;

- обмен информацией между органами муниципального управления;

- экономия бюджетных средств.

Автоматизация платежей ЖКХ связана с обеспечением эффективного трехстороннего взаимодействия между потребителями, агрегатором и поставщиками услуг ЖКХ.

Таким образом, тема ВКР «Разработка ПО для автоматизации денежных переводов поставщикам услуг» является актуальной.

Цель работы – разработка программного обеспечения автоматизации денежных переводов поставщикам услуг.

Для достижения цели в работе решены следующие задачи:

– постановка задачи на разработку;

– проектирование и моделирование программного обеспечения;

– разработка и тестирование программного обеспечения.

Объект – система оплаты услуг ЖКХ.

Предмет – программное обеспечение автоматизации денежных переводов поставщикам услуг ЖКХ.

Во введении обоснована актуальность темы, установлены цель, задачи, объект и предмет работы.

В первой главе выполнены постановка задачи на разработку ПО для автоматизации денежных переводов поставщикам услуг, формирование требований к ПО и сравнительный анализ используемых аналогов.

Во второй главе выполнено проектирование ПО для автоматизации денежных переводов поставщикам услуг, в ходе которого проведен выбор архитектуры ПО, разработаны логической модели ПО и алгоритм формирования перевода из платежей.

В третьей главе проведены выбор средств разработки, реализация и тестирование приложения.

В заключении представлены краткие результаты работы и дана оценка выполнения задач и достижения цели.

Использование результатов работы обеспечит эффективное формирование переводов поставщикам услуг ЖКХ.

1 Постановка задачи на разработку ПО для автоматизации денежных переводов поставщикам услуг

1.1 Постановка задачи

К числу особенностей ЖКХ как отрасли народного хозяйства можно отнести:

– сочетание производственных и непроизводственных функций, связанных с изготовлением материальных продуктов и оказанием услуг;

– особую социальную значимость, усиливающую необходимость государственного регулирования и контроля со стороны потребителей;

– сочетание коммерческих (ориентированных на достижение прибыли) и некоммерческих организаций;

– отрасль представлена как естественными монополиями (транспортировка энергии и жидкостей), так и отраслями, в которых возможна и необходима конкуренция (производство товаров и услуг);

– многообразие организационно-правового статуса предпринимательства (с образованием и без образования юридического лица) и форм собственности;

– сочетание крупного (производство энергии, водоканал, трубопроводные сети и т.д.) и малого бизнеса;

– рассредоточение центров оказания услуг соответственно системе расселения, что обуславливает особую роль местных органов самоуправления;

– особая значимость экологического и санитарно-эпидемиологического контроля;

– необходимость гарантированного обеспечения минимума услуг независимо от платежеспособности населения.

Этим определяется актуальность разработки приложения, которое позволило бы агрегатору платежей ЖКХ эффективно формировать переводы поставщикам услуг ЖКХ из платежей, поступивших от потребителей.

Важной проблемой при этом является обеспечение неделимости платежей при переводе поставщику – платеж потребителя может быть или полностью переведен поставщику или полностью находиться в непереведенных у агрегатора платежей.

1.2 Формирование требований к ПО для автоматизации денежных переводов поставщикам услуг

Для формирования требований к приложению мы будем использовать классификацию FURPS+.

Данная классификация требований разработана и успешно применяется для создания приложений и информационных систем в настоящее время [8], [4].

Основа любого приложения или системы, согласно классификации FURPS+, это её функционал, то есть то, для чего предназначена система, и какие возможности в ней реализованы.

Классификация требований к системе FURPS+ была разработана Робертом Грэйди (Robert Grady) из Hewlett-Packard и предложена в 1992 году. Сокращение FURPS расшифровывается так:

– Functionality, функциональность

– Usability, удобство использования

– Reliability, надежность

– Performance, производительность

– Supportability, поддерживаемость [7]

На основании функциональных требований строятся диаграммы вариантов использования.[6]

Требования к проектируемому приложению представлены в таблице 1. Приоритет реализации конкретных требований приложения представлен значениями от 1 до 5, где 1 – наивысший приоритет, 5 – наименьший/

Таблица 1 – Требования к приложению

Требование	Приоритет	Описание
Functionality		
1 Ведение справочников потребителей и поставщиков	1	Пользователь-сотрудник должен иметь возможность добавить, удалить или отредактировать информацию в базе данных по поставщикам и потребителям услуг
2 Ввод информации по платежам	1	Пользователь-потребитель должен иметь возможность ввести информацию по платежу, совершаемому в адрес определенного поставщика услуг
3 Просмотр истории платежей	2	Пользователь-потребитель должен иметь возможность просмотра истории платежей
4 Формирование перевода	1	Пользователь-сотрудник должен иметь возможность сформировать перевод поставщику как сумму платежей от потребителей.
5 Просмотр истории переводов	2	Пользователь-сотрудник должен иметь возможность просмотра истории переводов
Usability		
1 Удобный и понятный интерфейс	1	Каждое действие должно быть интуитивно понятным, используются стандартные жесты для управления приложением и логичные пункты в меню
Reliability		
1 Доступность системы 24/7		Приложение (хотя бы частично) должно быть доступно клиенту вне зависимости от внешних факторов (работа сервера, баз данных, интернет-соединение)
Performance		
1 Время запуска приложения не более 3-х секунд	3	-
2 Время отклика приложения не более 1 секунды	2	-
Supportability		
Корректная работа функционала на различных устройствах	1	Приложение должно стабильно работать на любых устройствах, соответствующих минимальным системным требованиям

1.3 Сравнительный анализ используемых аналогов

Для целесообразности разработки нового приложения необходимо проверить, если ли среди используемых аналогов то, что подходит под сформулированные требования. Наиболее популярными решениями в области передачи платежей ЖКХ являются приложения Квартплата.РФ, Инфокрафт , 1С: Председатель ТСЖ.

Результаты анализа аналогов приведены в таблице 2.

Таблица 2 – Сравнительный анализ аналогов

Функциональность	Квартплата.РФ	Инфокрафт	1С: Председатель ТСЖ
Прием платежей	+	-	-
История показаний приборов учёта	+	+	+
История платежей	+	+	-
Формирование перевода поставщику услуг ЖКХ из неделимых платежей потребителей	-	-	-

Ключевая функция разрабатываемого приложения – формирование перевода поставщику услуг ЖКХ из неделимых платежей потребителей в выбранных для анализа аналогах отсутствует.

Выводы по разделу:

Для проектируемого приложения сформированы требования по методологии FURPS+, указан приоритет реализации каждого требования.

Также был проведён анализ аналогов, показавший отсутствие на рынке готового приложения, соответствующего предъявляемым требованиям.

2 Проектирование ПО для автоматизации денежных переводов поставщикам услуг

2.1 Выбор архитектуры ПО для автоматизации денежных переводов поставщикам услуг

«Стандартным решением для организации связи Java-приложений с базой данных является JDBC. Именно эта библиотека входит в комплект JavaBeans и поставляется в комплекте средств разработки JDK. Эта библиотека обеспечивает Java-программам универсальное средство для работы с СУБД. JDBC работает аналогично ODBC - универсальному механизму доступа к базам данных. Вызовы методов JDBC могут преобразовываться в вызовы функций ODBC с помощью специального моста JDBC - ODBC. JDBC выполняет три основные функции работы с базами данных:

– установление связи с СУБД;

– передача базе данных SQL-запросов;

– обработка полученных результатов.

Архитектура JDBC подразумевает использование как минимум четырех частей: Java-приложения, которое вызывает методы JDBC; собственно компонента JDBC, который распределяет клиентские вызовы по соответствующим базам данных; драйвера JDBC, который поддерживает сеанс связи с конкретной базой данных, и базы данных. Если драйвер JDBC поставляется производителем базы данных, то Java-приложение может получить доступ к ней через механизм JDBC без посредников. Этот вариант JDBC позволяет работать в двухуровневой архитектуре клиент-сервер.

В целом, JDBC предлагает полный набор необходимых низкоуровневых функций для связи с базами данных, который может быть использован другими приложениями или трансляторами JSQL. Метод CallableStatement позволяет запускать внутренние процедуры СУБД, что

весьма важно в тех случаях, когда пользователям запрещено напрямую изменять содержание базы данных, но разрешено работать с определенными функциями СУБД. Этот метод позволяет более точно настроить права доступа пользователей к базе данных.» [14]

На основании изложенного для разрабатываемого программного обеспечения выбираем двухуровневую архитектуру.[2] Связь приложения-клиента с сервером базы данных будет реализована с помощью JDBC [5].

2.2 Разработка логической модели ПО для автоматизации денежных переводов поставщикам услуг

Приложение должно обеспечить возможность:

– редактирования справочника Поставщик услуг;

– редактирования справочника Потребитель услуг;

– ввода платежей через пользовательский интерфейс с указанием лицевого счета, суммы платежа и даты платежа;

– формирования перевода.

Формирование перевода выполняется следующим образом: через пользовательский интерфейс выбирается поставщик услуг, приложение отображает сумму платежей, не переведенных данному поставщику.

Через пользовательский интерфейс вводится сумма перевода. Приложение должно из всех непереведенных платежей собрать у перевода максимально близкую к указанной сумме перевода.

В итоге перевод должен сохраниться, переведенные платежи должны отметиться как переведенные.

На основании данного описания разработана диаграмма прецедентов (рисунок 1), содержащая двух акторов – сотрудника и потребителя. Сотрудник выполняет редактирование справочников и формирование переводов поставщикам. Потребитель выполняет ввод платежей.

Рисунок 1 – Диаграмма прецедентов

На основании диаграммы прецедентов разработана концептуальная модель данных (рисунок 2) разрабатываемого приложения. Модель выполнена в нотации Чена [10], [11].

Модель содержит следующие сущности:

– поставщик услуг (организация или предприятие, оказывающее услуги в сфере ЖКХ);

– потребитель услуг (физическое лицо, оплачивающее услуги ЖКХ);

– платеж (исходящая от потребителя услуг денежная транзакция);

– перевод денежных средств (входящая поставщику услуг денежная транзакция).

Связи в модели данных разрабатываемого приложения:

– поставщик услуг связан со многими потребителями услуг;

– потребитель услуг осуществляет множество платежей;

– поставщик услуг получает множество переводов.

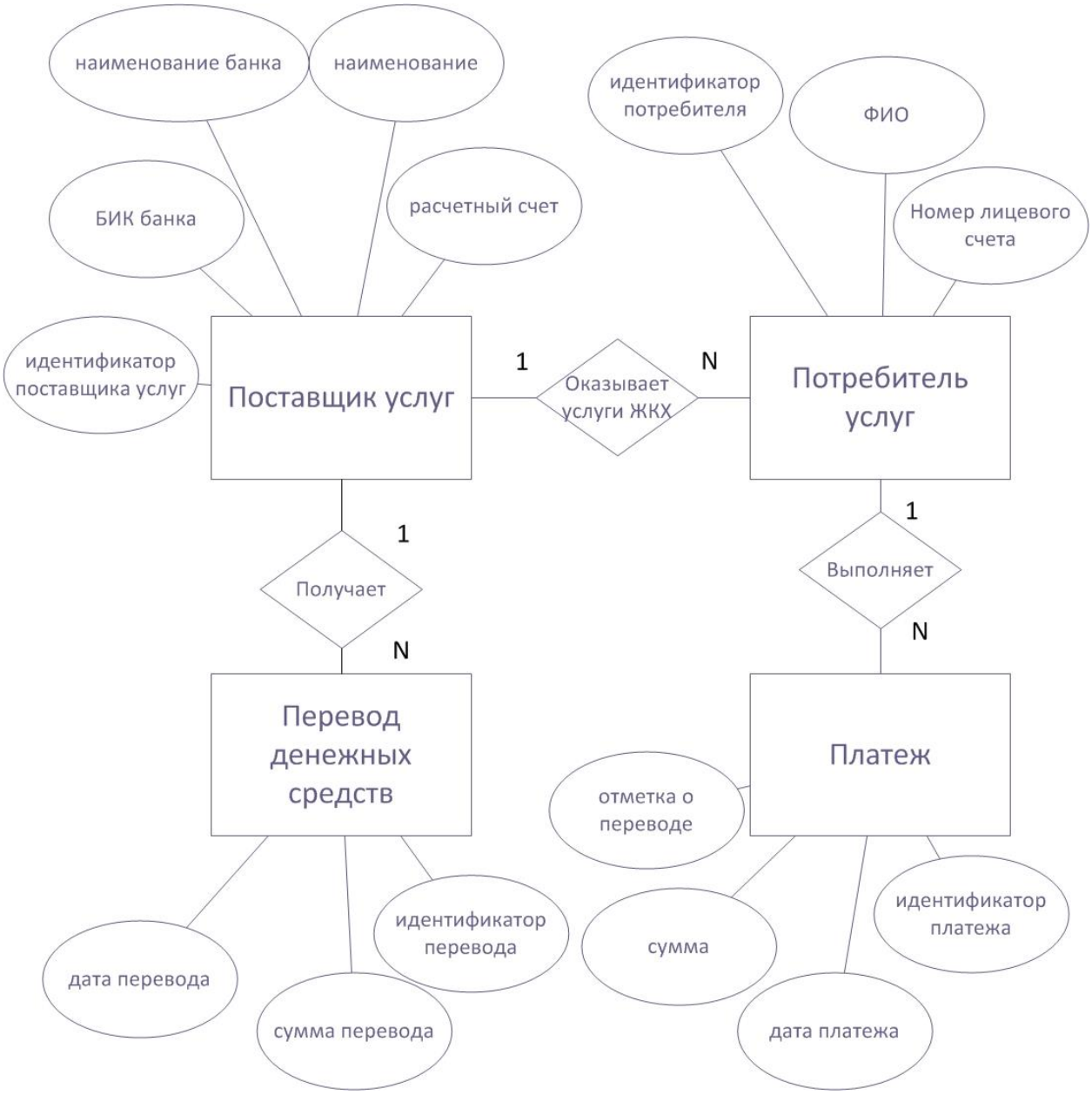

Рисунок 2 – Концептуальная модель данных

Сущности, связи и атрибуты концептуальной модели служат основой для логического и физического моделирования данных.

Логическая модель данных определяет тип базы данных (реляционная БД) показана на рисунке 3.

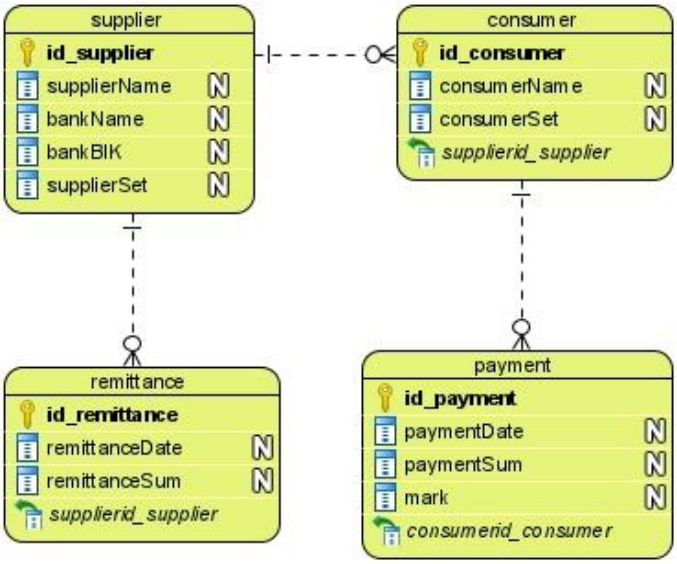

Рисунок 3 – Логическая модель данных

Физическая модель данных учитывает особенности конкретной СУБД (PostgreSQL) показана на рисунке 4.

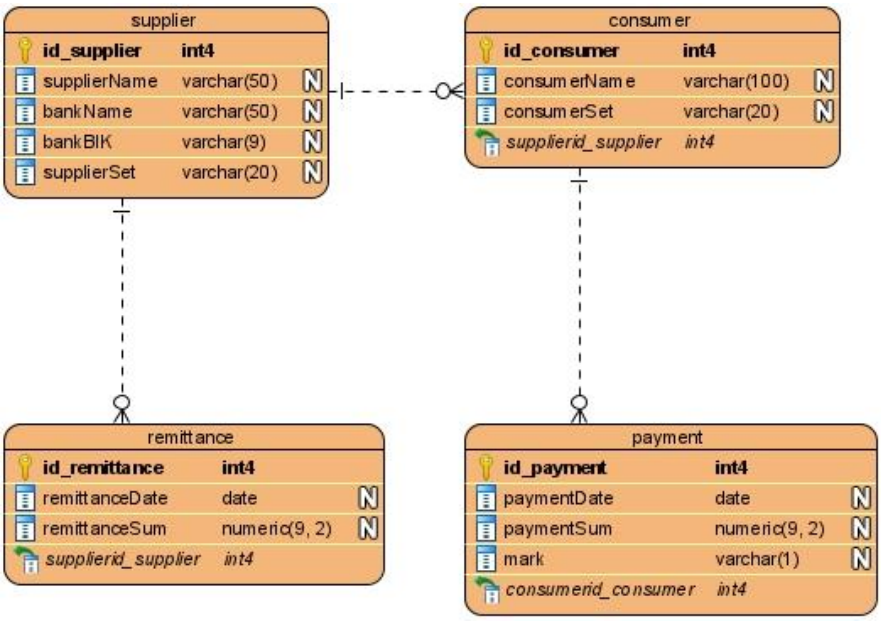

Рисунок 4 – Физическая модель данных

Физическая модель служит основой для создания структуры таблиц базы данных в выбранной СУБД PostgreSQL.

На базе физической модели в Visual Paradigm [20] сгенерирован SQL код для создания таблиц:

```
CREATE TABLE consumer (
  id_consumer         SERIAL NOT NULL,
  consumerName        varchar(100),
  consumerSet         varchar(20),
  supplierid_supplier int4 NOT NULL,
  PRIMARY KEY (id_consumer));
CREATE TABLE payment (
  id_payment          SERIAL NOT NULL,
  paymentDate         date,
  paymentSum          numeric(9, 2),
  mark                varchar(1) DEFAULT '0',
  consumerid_consumer int4 NOT NULL,
  PRIMARY KEY (id_payment));
CREATE TABLE remittance (
  id_remittance       SERIAL NOT NULL,
  remittanceDate      date,
  remittanceSum       numeric(9, 2),
  supplierid_supplier int4 NOT NULL,
  PRIMARY KEY (id_remittance));
CREATE TABLE supplier (
  id_supplier  SERIAL NOT NULL,
  supplierName varchar(50),
  bankName     varchar(50),
  bankBIK      varchar(9),
  supplierSet  varchar(20),
  PRIMARY KEY (id_supplier));
ALTER TABLE consumer ADD CONSTRAINT FKconsumer256028 FOREIGN
KEY (supplierid_supplier) REFERENCES supplier (id_supplier);
ALTER TABLE payment ADD CONSTRAINT FKpayment327107 FOREIGN
KEY (consumerid_consumer) REFERENCES consumer (id_consumer);
ALTER TABLE remittance ADD CONSTRAINT FKremittance529365
FOREIGN KEY (supplierid_supplier) REFERENCES supplier
(id_supplier);
```

Данный скрипт обеспечит создание в pgAdmin [18] таблиц и связей соответствующих физической модели.

2.3 Разработка алгоритма формирования перевода из платежей

Формирования перевода из платежей состоит в том, что пользователь вводит целевую сумму, которую необходимо перевести поставщику услуг. Затем из таблицы платежей формируется набор, сумма которого (сумма подбора) равна или максимально близка к целевой сумме. Перевод совершается, о чем вносится запись в таблице переводов. Платежи, участвующие в формировании суммы подбора, отмечаются в таблице платежей в поле «Отметка о переводе» и в дальнейшем в формировании переводов не используются.

Очевидно, что добавление записи в таблицу переводов и изменение поля «Отметка о переводе» таблицы платежей должно выполниться транзакцией – действие должно быть атомарным и выполняться полностью или не выполняться совсем.

В основе разрабатываемого алгоритма лежит задача Combination Sum 2. Задача формулируется следующим образом «Имея набор номеров-кандидатов (кандидаты) и целевое число (цель), найдите все уникальные комбинации в кандидатах, где сумма номеров-кандидатов равна целевому. Каждое число в кандидатах может использоваться только один раз в комбинации. В наборе решений не должно быть повторяющихся комбинаций.»[12]

Текстовое описание алгоритма решения задачи Combination Sum 2 приведено на следующем листинге псевдокода [15]:

```
candidates - массив (вектор)
target - целое
result - 2D массив результатов (на старте пустой)
current - текущий массив (вектор)(на старте пустой)
n - счетчик n=0
sumTillNow - сумма элементов вектора current, sumTillNow=0
ВЫЗВАТЬ ФУНКЦИЮ combinationSum2Util(result, candidates,
current, n, sumTillNow, target)
ВЕРНУТЬ result
     функция combinationSum2Util
     - if sumTillNow == target
```

```
    // добавить массив current в массив result
    - result.push_back(current)
  - if sumTillNow > target
    - return
  -  prev = -1
  - loop for i = n; i <= candidates.size() - 1; i++
    - if prev != candidates[i]
       // добавить ій элемент candidates в current
       - current.push_back(candidates[i])
       - sumTillNow += candidates[i]
       // Рекурсивный вызов функции
        -   combinationSumUtil(result,   candidates,   i,
        target, sumTillNow, current)
       - sumTillNow -= current[current.size() - 1]
       // удалить последний элемент из массива current
       - current.pop_back()
       - prev = candidates[i]
    - end of if
```

Из анализа приведенного выше псевдокода применительно к решаемой задаче формирования перевода установлено:

– нет необходимости находить все уникальные комбинации элементов массива candidates дающие в сумме target, поэтому поиск можно остановить после первой найденной комбинации и вернуть ее в качестве решения;

– условие выхода из рекурсивной функции с сформированием результата if sumTillNow==target представляет собой точное сравнение и определяет целочисленный тип данных. В соответствии с физической моделью суммы в базе данных хранятся в формате numeric (9,2) – точный тип данных, предназначенный для финансовых расчетов с максимальным возможным значением порядка 9 миллионов рублей с округлением до копеек. Массив candidates имеет тип данных integer и хранит суммы в копейках. В соответствии с максимальным значением типа данных 2 147 483 647 в копейках может быть размещена сумма порядка 21 миллиона рублей. Таким образом, типы данных в базе и в расчетном модуле соответствуют друг другу;

– поскольку при отсутствии в candidates элементов, дающих в сумме target необходимо найти ближайшее решение, то при выходе по if sumTillNow > target необходимо обеспечить повторный расчет при изменённой целевой сумме target2 = target+1, затем, при отсутствии решения, при target2 = target-1, target2 = target+2 и далее до нахождения решения.

В псевдокоде алгоритм подбора платежей для перевода имеет вид:

```
candidates - массив (вектор)
target - целевая сумма
target2 - скорректированная целевая сумма(на старте target2=target)
 result - массив результатов (вектор) (на старте пустой)
 current - текущий массив (вектор)(на старте пустой)
 n - счетчик n=0
 sumTillNow - сумма элементов вектора current sumTillNow=0
 ВЫЗВАТЬ ФУНКЦИЮ combinationSum2Util(result, candidates, current, n, sumTillNow, target)
ВЕРНУТЬ result
     функция combinationSum2Util function
     - if sumTillNow == target2
       // добавить массив current в массив result
       - result.push_back(current)
       - return
     - if sumTillNow > target
          loop for j = j; j <= target - 1; i++
          if j четное
          target2 = target2 - j;
          else
          target2 = target2 = j;
          // Рекурсивный вызов функции
          - combinationSumUtil
     -  prev = -1
     - loop for i = n; i <= candidates.size() - 1; i++
       - if prev != candidates[i]
          // добавить ій элемент candidates в current
          - current.push_back(candidates[i])
          - sumTillNow += candidates[i]
          // Рекурсивный вызов функции
          - combinationSumUtil
          - sumTillNow -= current[current.size() - 1]
          // удалить последний элемент из массива current
          - current.pop_back()
          - prev = candidates[i]
       - end of if
```

Общая схема алгоритма формирования перевода показана на рисунке 5.

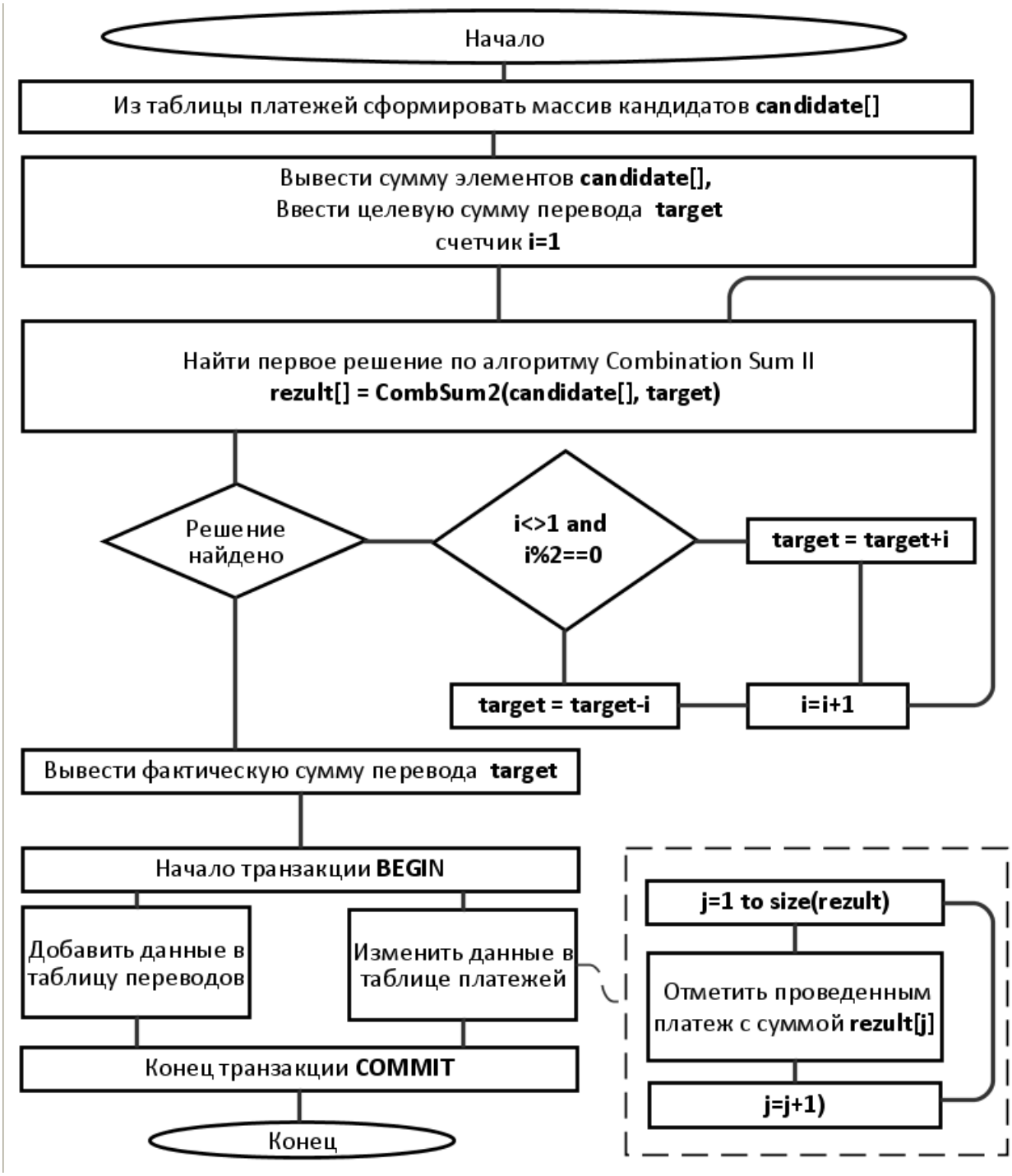

Рисунок 5 – Схема алгоритма формирования перевода

В результате работы алгоритма выводится массив rezult, содержащий величины платежей из таблицы платежей, сумма которых равна target. При изменении поля mark таблицы payment запросом вида

```
UPDATE payment
     SET mark = '1'
```

```
WHERE mark<>’1’ AND paymentSum = 100.00;
```

будет иметь место нарушение логической целостности данных – например, если в таблице присутствует несколько необработанных платежей по 100 рублей, то они все будут отмечены как обработанные, а должен быть отмечен только один платеж. Для ограничения количества записей, на котором выполнится запрос, в SQL используется команда LIMIT, но данная команда в диалекте SQL Postgres работает только в запросах на выборку SELECT [16]. Поэтому в запросе на изменение необходимо использовать вложенный запрос на выборку, возвращающий единственную запись по суррогатному ключу:

```
UPDATE payment
     SET mark = '1'
     WHERE id_payment =
          ( SELECT id_payment
                    FROM payment
                    WHERE paymentSum = 100.00
                          AND
                          mark <> ’1’
                    LIMIT 1 );
```

Такой запрос обеспечит изменение единственной записи вне зависимости от того, сколько записей соответствует условиям.

Выводы по разделу:

Во втором разделе выполнено обоснование выбора двухуровневой клиент-серверной архитектуры, проведено логическое моделирование программного обеспечения, разработаны диаграмма прецедентов, концептуальная, логическая и физическая модели данных, разработан алгоритм формирования перевода из платежей.

3 Реализация ПО для автоматизации денежных переводов поставщикам услуг

3.1 Выбор средств разработки

При выборе средств разработки необходимо выбрать язык программирования, СУБД, IDE и, при необходимости, дополнительные средства для разработки и тестирования приложения.

«Java — язык программирования, который актуален уже почти 20 лет. Созданный Джеймсом Гослингом в 1995 году, он до сих пор входит в тройку лидеров многих рейтингов по популярности и востребованности.

Это язык программирования общего назначения и первый компилируемо-интерпретируемый язык. Благодаря Java Virtual Machine (JVM) код, написанный на этом языке, можно запускать в любой среде.

В Java много внимания уделено раннему обнаружению ошибок и динамической проверке во время работы программы. Поэтому язык считается безопасным и на нем часто пишут важные системы: банковские терминалы, системы обработки транзакций, сервисы координации перелетов и другие. Кроме того, Java достаточно дешевый в обслуживании — запускать и работать с ним можно практически с любого компьютера, вне зависимости от конкретной аппаратной инфраструктуры.» [9]

В качестве языка программирования выбрана Java SDK 8.0.2.

«MySQL - Это одна из самых популярных реляционных баз данных. Первоначально выпущенная как решение с открытым исходным кодом, MySQL теперь принадлежит корпорации Oracle. Написанный на C и C++ MySQL, хорошо работает с такими платформами, как Linux, Windows, macOS, IRIX и другими.

Плюсы MySQL:

– версию MySQL community edition (для сообщества) можно установить бесплатно;

– простой синтаксис и умеренная сложность;

– совместимость с облачными решениями.

Минусы MySQL:

– проблемы масштабируемости - MySQL не был создан с учетом масштабируемости;

– не полностью открытый исходный код - частично MySQL находится под лицензией Oracle;

– ограниченное соответствие стандартам SQL - MySQL не поддерживает некоторые стандартные функции SQL., а с другой стороны, MySQL имеет некоторые расширения и отдельные функции, не соответствующие стандартам SQL.»[1]

«СУБД PostgreSQL - разделяет свою популярность с MySQL. Это объектно-реляционная СУБД, в которой пользовательские объекты и табличные подходы объединяются для создания более сложных структур данных. Кроме того, PostgreSQL имеет много общего с MySQL. Он направлен на укрепление стандартов соответствия и расширяемости. Следовательно, он может обрабатывать любую рабочую нагрузку, как для продуктов с одной машиной, так и для сложных приложений. Принадлежащий и разработанный Глобальной группой разработчиков PostgreSQL, он по-прежнему остается полностью открытым исходным кодом. Эта СУБД доступна для использования с такими платформами, как Linux, Microsoft Windows, iOS, Android и многими другими. Существует платная версия и русскоязычная техподдержка.

Плюсы PostgreSQL:

– система надёжно работает с большими объёмами данных. Зачастую без последствий проходят внештатные ситуации типа аварийного отключения питания;

– отличная масштабируемость. Вертикальная масштабируемость является отличительной чертой PostgreSQL, в отличие от СУБД MySQL. Учитывая, что почти любое пользовательское программное решение имеет

тенденцию к росту, что приводит к расширению базы данных, этот конкретный вариант, безусловно, поддерживает рост и развитие бизнеса;

– легко интегрируемые сторонние инструменты. Система управления базами данных PostgreSQL имеет мощную поддержку дополнительных инструментов, как бесплатных, так и коммерческих;

– поддержка с открытым исходным кодом и на основе сообщества;

– высокопроизводительные и надёжные механизмы транзакций и репликации.

Минусы PostgreSQL:

– противоречивая документация. Хотя PostgreSQL имеет большое сообщество и оказывает сильную поддержку своим участникам, документации по-прежнему не хватает последовательности и полноты. Поскольку сообщество PostgreSQL довольно распределено, документация не соответствует одинаковым стандартам.

– слабо развиты инструменты отчетности и аудита.»[2]

«СУБД MS SQL-сервер является полностью коммерческим инструментом и одной из самых популярных реляционных СУБД. Он хорошо справляется с эффективным хранением, изменением и управлением реляционными данными. Для взаимодействия с базами данных SQL Server инженеры баз данных обычно используют язык Transact-SQL (T-SQL), который является расширением стандарта SQL.

Плюсы MSSQL

– разнообразие версий;

– комплексное решение для обработки бизнес-данных..

– исчерпывающая документация;

– поддержка облачных решений.

Минусы MS SQL:

– привязка к платформе Microsoft Windows;

– высокая стоимость;

– высокая требовательность к аппаратным ресурсам;.

– сложный процесс настройки;

– восстановление данных после аварийного отключения питания обязательно требует участия специалиста.»[2]

В качестве СУБД выбрана PoctgreSQL 9.5.25.

«Интегрированная среда разработки или по-английски Integrated development environment — IDE — это программа, которая содержит в себе инструменты для разработки программного обеспечения. Обычно современная среда разработки включает в себя:

– текстовый редактор с подсветкой кода;

– компилятор или интерпретатор;

– браузер классов, инспектор объектов и диаграмму иерархии классов;

– средства автоматизации сборки;

– отладчик;

– средства для интеграции с системами управления версиями (Git);

– инструменты для упрощения конструирования графического интерфейса пользователя.»[3]

« IDE IntelliJ IDEA с точки зрения возможностей и цены поставляется в двух вариантах: бесплатного Community edition, и платного Ultimate edition с расширенной функциональностью.

Community edition предназначена для JVM- и Android-разработки. Бесплатная версия поддерживает Java, Kotlin, Groovy и Scala; Android; Maven, Gradle и SBT; работает с системами контроля версий Git, SVN, Mercurial и CVS.

Ultimate edition приспособлена для веб- и enterprise-разработки»[3].

«IDE Eclipse долгие годы уверенно держала пальму первенства среди Java IDE. Эта среда полностью бесплатная, с открытым исходным кодом, написанным преимущественно на Java. Тем не менее, её модульная архитектура позволяет использовать Eclipse и с другими языками.

Портативность Java помогает Eclipse быть кроссплатформенной средой: эта IDE работает на Linux, Mac OS X, Solaris и Windows.

Eclipse работает довольно медленно. Часть расходов ресурсов Eclipse можно отнести на счёт её встроенного инкрементного компилятора, который запускается всякий раз при загрузке файла или обновлении кода.

Независимо от сборки, проект Eclipse поддерживает модель контента, которая содержит информацию об иерархии типов, ссылок и объявлениях Java-элементов.»[3].

«NetBeans IDE представляет собой бесплатную интегрированную среду разработки с открытым кодом для разработчиков программного обеспечения. Среда IDE NetBeans предоставляет все средства, необходимые для создания профессиональных приложений рабочей среды, корпоративных, мобильных и веб-приложений на языках Java, C/C++, а также на других динамических языках. Среда IDE может работать на различных платформах, включая операционные системы Windows, Linux, Solaris и Mac, отличается простотой установки и удобством использования и не требует дополнительной настройки. Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведётся независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org. В сентябре 2016 года Oracle передала интегрированную среду разработки NetBeans в руки фонда Apache.»[17].

В качестве IDE выбрана NetBeans 8.

В качестве средств разработки выбрано СУБД PoctgreSQL 9.5.25, IDE NetBeans 8, java SDK 8.0.2.

3.2 Реализация приложения

База данных remittance и соответствующая структура таблиц и связей создана средствами pgAdmin, входящей в состав PoctgreSQL с использованием SQL запроса приведенного в пункте 2.2.

Проект разрабатываемого приложения RemittanceProject создан в NetBeans 8 на основании шаблона Java application.

В соответствии с поставленными задачами взаимодействие приложения с базой данных реализует три основных прецедента:

– ведение справочников;

– ввод платежей;

– формирование переводов.

Поскольку эти действия будут выполняться разными пользователями и в разное время, то для каждого из них выполнена отдельная форма, соответственно для ведения справочников, ввода платежа и формирования перевода.

Форма ведения справочников представлена на рисунках 6 и 7. В основу положена форма RemittanceMain класса JFrameForm, позволяющая быстро и эффективно разработать пользовательский интерфейс с применением визуального конструктора встроенной библиотеки Swing. Ведение дочерней таблицы Потребители выполняется с помощью выбора соответствующего поставщика из раскрывающегося списка - элемента ComboBox.

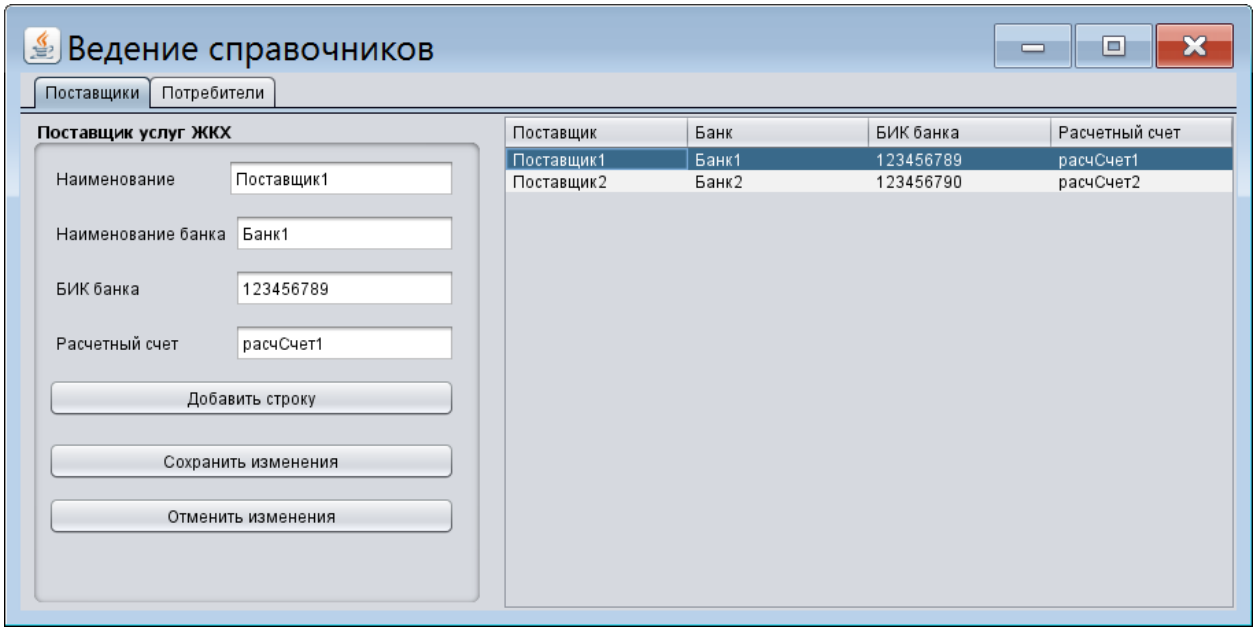

Рисунок 6 – Форма ведения справочников. Вкладка Поставщики

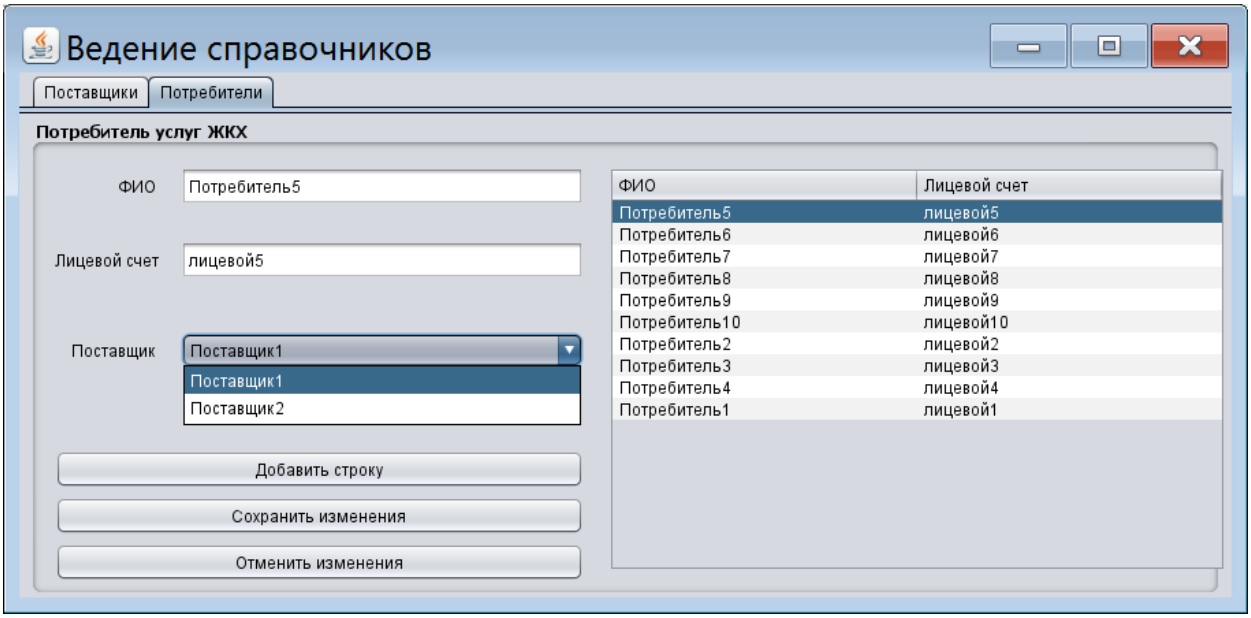

Рисунок 7 – Форма ведения справочников. Вкладка Потребители

Подключение remittanceDB базы данных к проекту [13] выполнено с использованием адреса jdbc:postgresql://localhost:5432/remittance и драйвера postgresql-9.2-1002.jdbc4.jar кодом

```
    Class.forName("org.postgresql.Driver");
    String url = "jdbc:postgresql://localhost:5432/remittance";
    Connection    conn    =    DriverManager.getConnection(url,"
postgres", " postgres");
```

Для связывания элементов формы с данными созданы два объектных класса Supplier и Consumer соответствующие таблицам базы данных [19].

Интерфейс TableModel позволяет приложению управлять данными в JTable объекте. Конструкторы ConsumerTableModel.java и SupplierTableModel.java реализуют этот интерфейс. Они указывают, как JTable объекты должны извлекать данные из RowSet объектов и отображать их в таблицах.

Перед реализацией методов интерфейса TableModel конструкторы класса ConsumerTableModel и SupplierTableModel инициализируют

различные переменные-члены, необходимые для этих методов. Конструктор касса ConsumerTableModel:

```
public ConsumerTableModel(CachedRowSet rowSetArg)
    throws SQLException {
    this. consumerRowSet = rowSetArg;
    this.metadata = this.consumerRowSet.getMetaData();
    numcols = metadata.getColumnCount();
    // Получить количество строк
    this.consumerRowSet.beforeFirst();
    this.numrows = 0;
    while (this.consumerRowSet.next()) {
        this.numrows++;
    }
    this.consumerRowSet.beforeFirst();
}
```

В данном конструкторе инициализируются следующие переменные-члены:

– CachedRowSet consumerRowSet: сохраняет содержимое таблицы Consumer;

– ResultSetMetaData metadata: извлекает количество столбцов в таблице Consumer, а также имена каждого из них;

– int numcols, numrows: сохраняет количество столбцов и строк в таблице Consumer.

Класс ConsumerFrame реализует метод rowChanged из интерфейса RowSetListener:

```
public void rowChanged(RowSetEvent event) {
    CachedRowSet currentRowSet =
        this.myConsumerTableModel. consumerRowSet;
    try {
        currentRowSet.moveToCurrentRow();
        ConsumerTableModel = new ConsumerTableModel(
            ConsumerTableModel.getConsumerRowSet());
        table.setModel(ConsumerTableModel);
    } catch (SQLException ex) {
        JDBCTutorialUtilities.printSQLException(ex);
        // Отобразить ошибку в диалоговом окне
        JOptionPane.showMessageDialog(
            ConsumerFrame.this,
            new String[] {
                // Отобразить двухстрочное сообщение
                ex.getClass().getName() + ": ",
```

```
                    ex.getMessage()
                }
            );
        }
    }
```

Этот метод вызывается, когда пользователь добавляет строку в таблицу.

Кнопка Добавить строку осуществляет добавление в справочник нового поставщика или потребителя. При этом в таблицу добавляется строка в которую записываются данные из текстовых полей ввода. При добавлении нового потребителя в качестве поставщика ему записывается значение из поля со списком.

Код для добавления записи в таблицу потребителей приведен ниже.

```
button_ADD_ROW.addActionListener(
     new ActionListener() {
     public void actionPerformed(ActionEvent e) {
     JOptionPane.showMessageDialog(
          ConsumerFrame.this,new String[] {
               "Adding the following row:",
               "Потребитель: [" +
               textField_ consumerName.getText() +
               "]",
               "Лицевой счет: [" +
               textField_ consumerSet.getText() + "]",
               "Поставщик:  ["  +  supplierName.getText()  +
               "]",
          }
     );
     try {
          ConsumerTableModel.insertRow(
            textField_ consumerName.getText(),
            textField_ consumerSet.getText(),
            Integer.parseInt(
            textField_supplierid_supplier.getText().trim()
          );
    } catch (SQLException sqle) {
       displaySQLExceptionDialog(sqle);
    }
   }
});
```

Когда пользователь нажимает эту кнопку, он создает диалоговое окно сообщения, в котором отображается строка, которую необходимо добавить в таблицу, вызывает метод ConsumerTableModel.insertRow, который добавляет строку в переменную-член ConsumerTableModel.consumerRowSet.

Если SQLException сброшено, метод ConsumerFrame.displaySQLExceptionDialog создает диалоговое окно сообщения, в котором отображается содержимое файла SQLException.

Форма ввода платежа (рисунок 8) обеспечивает внесение данных в таблицу платежей - указывается номер лицевого счета, сумма платежа и дата платежа. После ввода лицевого счета отображается ФИО потребителя, формируется таблица с историей платежей, содержащая даты и суммы предыдущих платежей на данный лицевой счет, и появляется возможность ввести дату и сумму нового платежа.

При нажатии кнопки Оплатить данные добавляются в таблице платежей. При нажатии кнопки Отменить все поля формы очищаются.

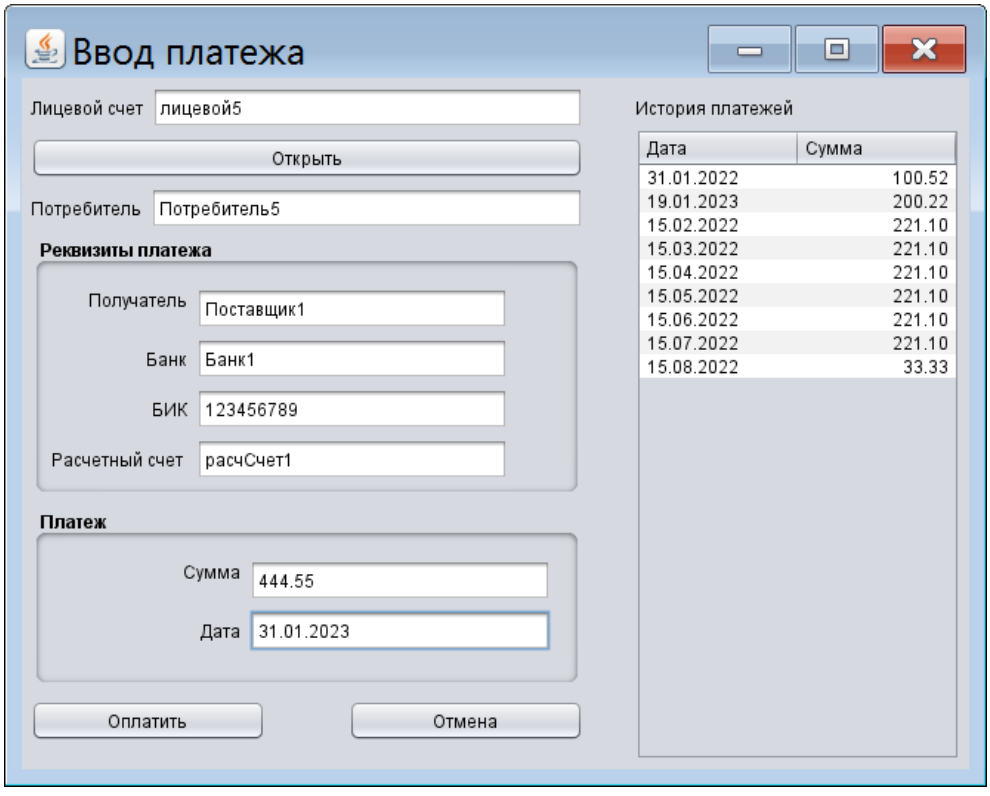

Рисунок 8 – Форма ввода платежа

Форма формирования перевода (рисунок 9) обеспечивает выбор поставщик услуг из раскрывающегося списка. Затем приложение отображает

сумму платежей, не переведенных данному поставщику и таблицу с историей переводов. Вводится желательная сумма перевода – целевая сумма. При нажатии кнопки Сформировать перевод запускается подбор платежей формирующий из всех непереведенных платежей сумму перевода максимально близкую к указанной сумме перевода.

На форме отображается найденная сумма платежей – фактическая сумма перевода.

При нажатии кнопки Перевести в таблицу переводов заносится новая запись и в таблице платежей поле Отметка о переводе отмечается Истиной для строк которые участвовали в формировании суммы перевода.

При нажатии кнопки Отмена все поля формы очищаются.

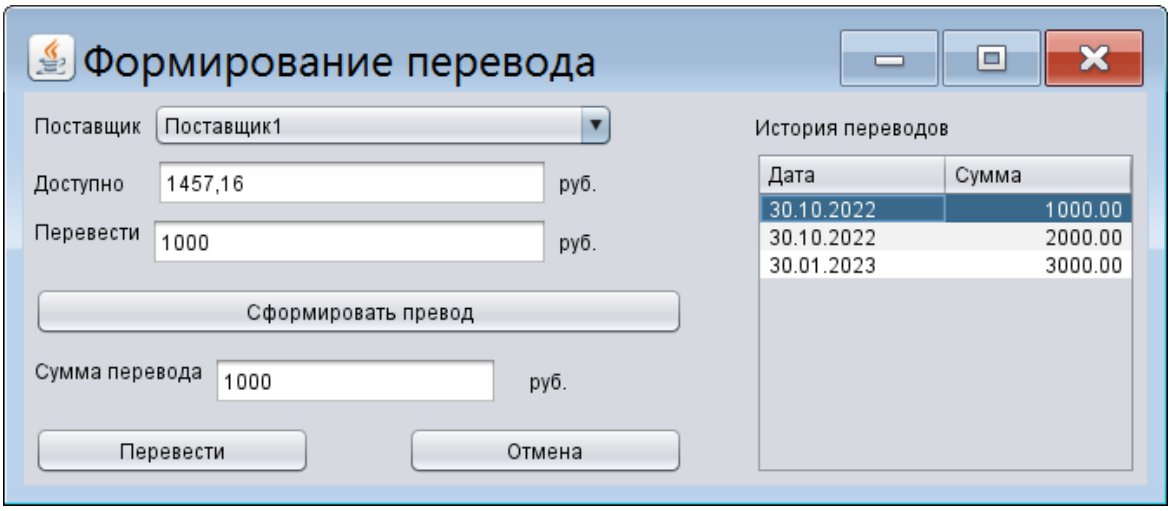

Рисунок 9 – Форма формирования перевода

Сборка перевода выполняется программным модулем, разработанным на основе псевдокода, показанного в пункте 2.3:

```
    public List<List<Integer>> combinationSum2(int[] candidates,
int target) {
        List<List<Integer>>        result        =        new
ArrayList<List<Integer>>();
        List<Integer> curr = new ArrayList<Integer>();
        Arrays.sort(candidates);
```

```
        helper(result, curr, 0, target, candidates);
        return result;
    }
    public void helper(List<List<Integer>> result, List<Integer>
curr, int start, int target, int[] candidates){
        if(target==0){
            result.add(new ArrayList<Integer>(curr));
            return;
        }
        if(target<0){
            return;
        }
        int prev=-1;
        for(int i=start; i<candidates.length; i++){
            if(prev!=candidates[i]){
                curr.add(candidates[i]);
                helper(result, curr, i+1, target-candidates[i],
candidates);
                curr.remove(curr.size()-1);
                prev=candidates[i];
            }
        }
    }
```

Разработанное приложение обеспечивает требуемую функциональность – ведение справочников, ввод платежей и формирование переводов и обладает удобным пользовательским интерфейсом. На следующем этапе необходимо выполнить тестирование приложения.

3.3 Тестирование приложения

За основу плана тестирования взяты тестовые данные задачи Combination Sum 2 [12]:

– кандидаты 10, 1, 2, 7, 6, 1, 5;

– цель 8,

– решение 1, 7; 1, 2, 5; 2, 6; 1, 1, 6.

Разработанное программное обеспечение обладает рядом отличий от типового примера:

– ищется одно, а не все решения;

– в случае если точное решение не найдено выполняется подбор ближайшего решения;

– уже проведенные платежи не должны участвовать в расчете;

– платежи другому поставщику не должны участвовать в расчете;

– платежи от другого потребителя должны участвовать в расчете;

– из множества платежей на одну сумму при совершении перевода должно отмечаться только то количество платежей на эту сумму, которое присутствует в решении, при этом не должны затрагиваться платежи другому поставщику.

Тестовые данные в рассматриваемом случае будут иметь вид:

– платежи 10,00, 1,00, 2,00, 7,00, 6,00, 1,00, 5,00, 1,00 (проведен), 1,00 (другому поставщику), 1,00 (другого потребителя) ;

– сумма непереведенных платежей поставщику 9,00;

– целевая сумма перевода 8,10;

– решение [1,00 7,00];

– фактическая сумма перевода 8,00;

В программе явно не указывается какой из нескольких платежей с одинаковой суммой необходимо отмечать как проведенный. Поэтому после совершения перевода таблица платежей может быть в одном из трех возможных состояний:

– изменен статус второго платежа: 10,00, 1,00 (проведен), 2,00, 7,00 (проведен), 6,00, 1,00, 5,00, 1,00 (проведен), 1,00 (другому поставщику), 1,00 (другого потребителя);

– изменен статус шестого платежа: 10,00, 1,00, 2,00, 7,00 (проведен), 6,00, 1,00 (проведен), 5,00, 1,00 (проведен), 1,00 (другому поставщику), 1,00 (другого потребителя);

– изменен статус платежа от другого потребителя: 10,00, 1,00, 2,00, 7,00 (проведен), 6,00, 1,00, 5,00, 1,00 (проведен), 1,00 (другому поставщику), 1,00 (проведен, другого потребителя).

При повторном вызове формы формирования перевода сумма непереведенных платежей поставщику должна быть равна 1,00;

Дополнительно в базу данных внесена информация о предыдущих переводах для отображения на форме. Исходное состояние базы данных представлено в таблицах 3-6.

Таблица 3 - Исходное состояние таблицы поставщиков

Id_Supplier	SupplierName	BankName	BankBIK	SupplierSet
1	Поставщик1	Банк1	123456789	расчСчет1
2	Поставщик2	Банк2	123456790	расчСчет2

Таблица 4 - Исходное состояние таблицы платежей

Id_Payment	PaymentDate	PaymentSum	Mark	Consumerid_Consumer	Примечание
1	01.01.2023	10.00	0	1	Набор исходных для Combination Sum 2
2	02.01.2023	1.00	0	1	Набор исходных для Combination Sum 2
3	03.01.2023	2.00	0	1	Набор исходных для Combination Sum 2
4	04.01.2023	7.00	0	1	Набор исходных для Combination Sum 2
5	05.01.2023	6.00	0	1	Набор исходных для Combination Sum 2
6	06.01.2023	1.00	0	1	Набор исходных для Combination Sum 2
7	07.01.2023	5.00	0	1	Набор исходных для Combination Sum 2
8	08.01.2023	1.00	1	1	Платеж проведен
9	09.01.2023	1.00	0	2	Другой потребитель другому поставщику
10	10.01.2023	1.00	0	3	Другой потребитель этому поставщику

Таблица 5 - Исходное состояние таблицы потребителей

Id_Consumer	ConsumerName	ConsumerSet	Supplierid _Supplier	Примечание
1	Потребитель1	лицевой1	1	Основной потребитель
2	Потребитель2	лицевой2	2	Потребитель другого поставщика
3	Потребитель3	лицевой3	1	Другой потребитель этого поставщика

Таблица 6 - Исходное состояние таблицы переводов

Id_Remittance	RemittanceDate	RemittanceSum	Supplierid_supplier
1	15.06.2022	1 000.00	1
2	15.07.2022	2 000.00	1
3	15.08.2022	3 000.00	1
4	15.09.2022	1 000 000.00	2
5	15.10.2022	5 000.00	1

Данные сформированы в программе MS Excel импортированы в базу данных. В ходе тестирования с использованием пользовательского интерфейса добавлены новый поставщик, новый потребитель и новый платеж от нового потребителя новому поставщику на сумму 555,55.

Ввод нового поставщика показан на рисунке 10. При этом в таблицу поставщиков добавлена новая запись.

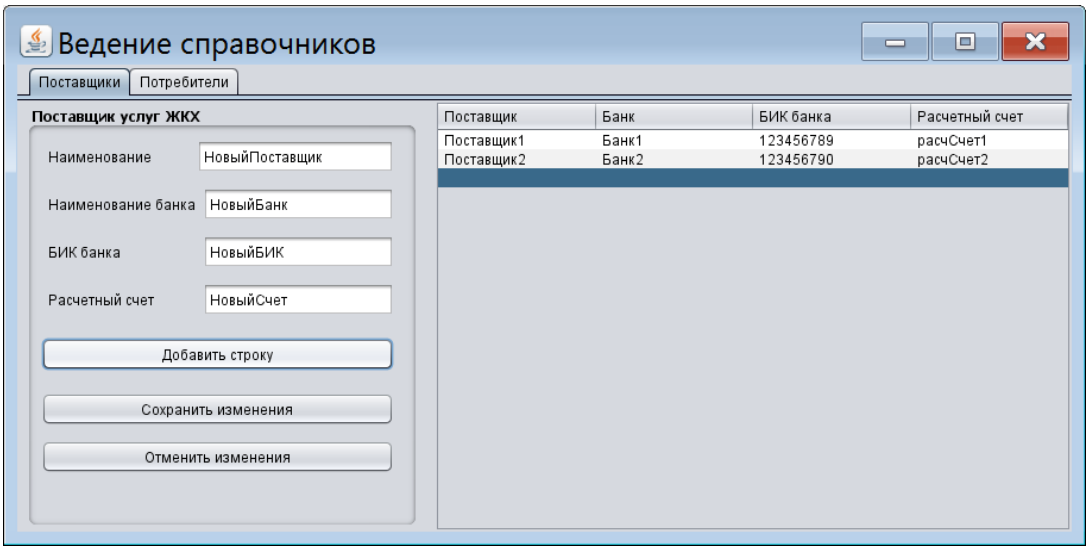

Рисунок 10 - Ввод нового поставщика

Ввод нового потребителя показан на рисунке 11. При этом в таблицу потребителей добавлена новая запись

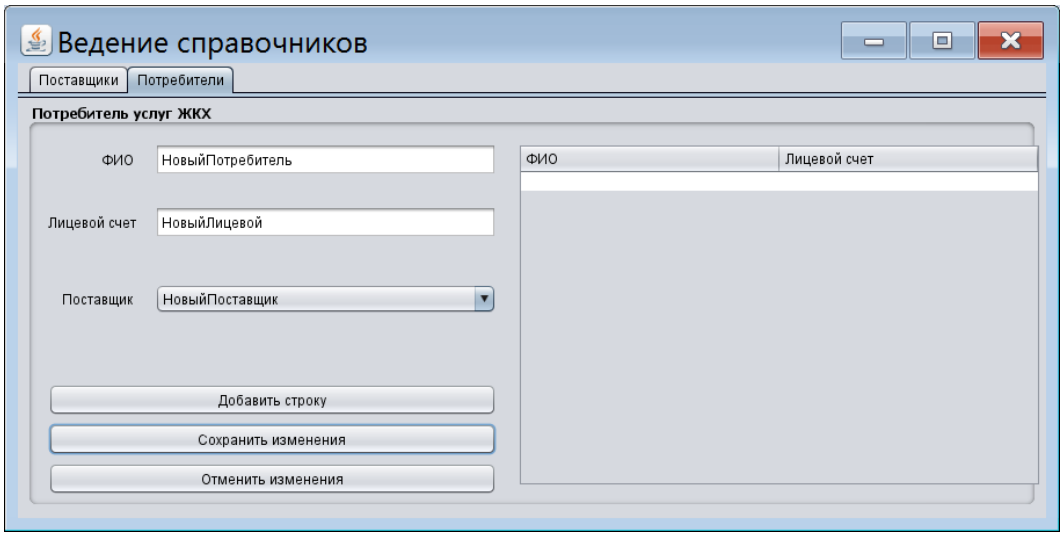

Рисунок 11 - Ввод нового потребителя

Ввод нового платежа показан на рисунке 12. При этом в таблицу платежей добавлена новая запись

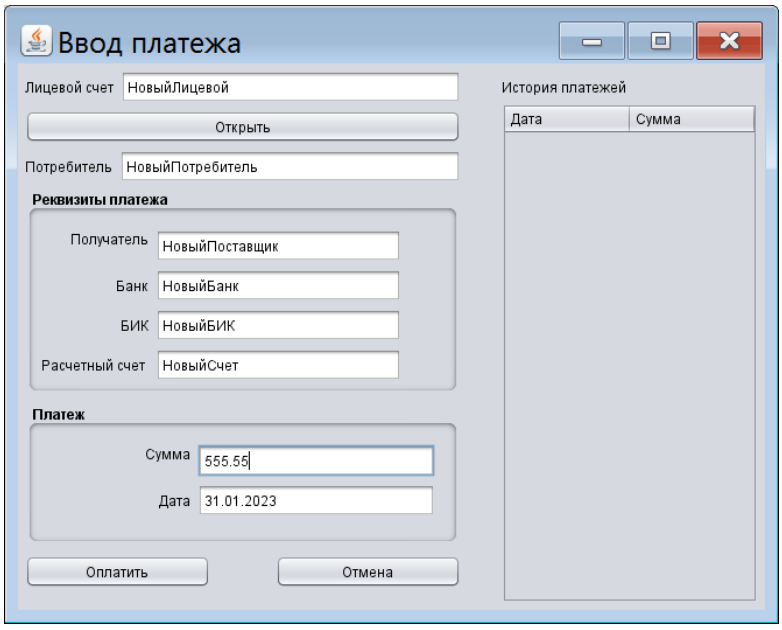

Рисунок 12 - Ввод нового платежа

Ввод нового перевода показан на рисунке 13. При этом в таблицу переводов добавлена новая запись, а в таблице платежей изменены 2 записи.

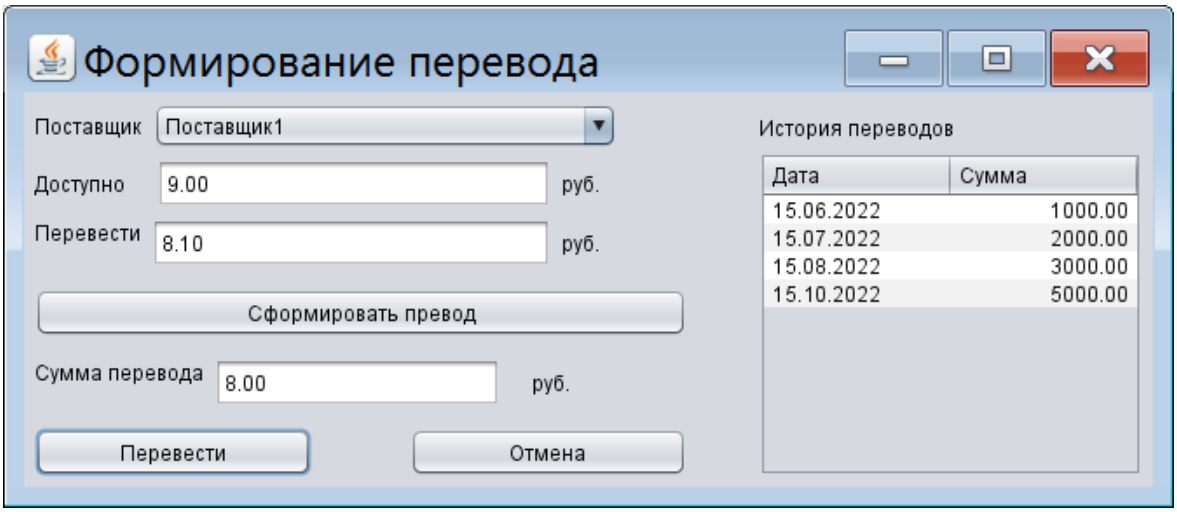

Рисунок 13 - Ввод нового перевода

При повторном вызове страницы формирования перевода (рисунок 14) сделанный перевод отобразился в списке переводов, а сумма непереведенных платежей соответственно уменьшилась.

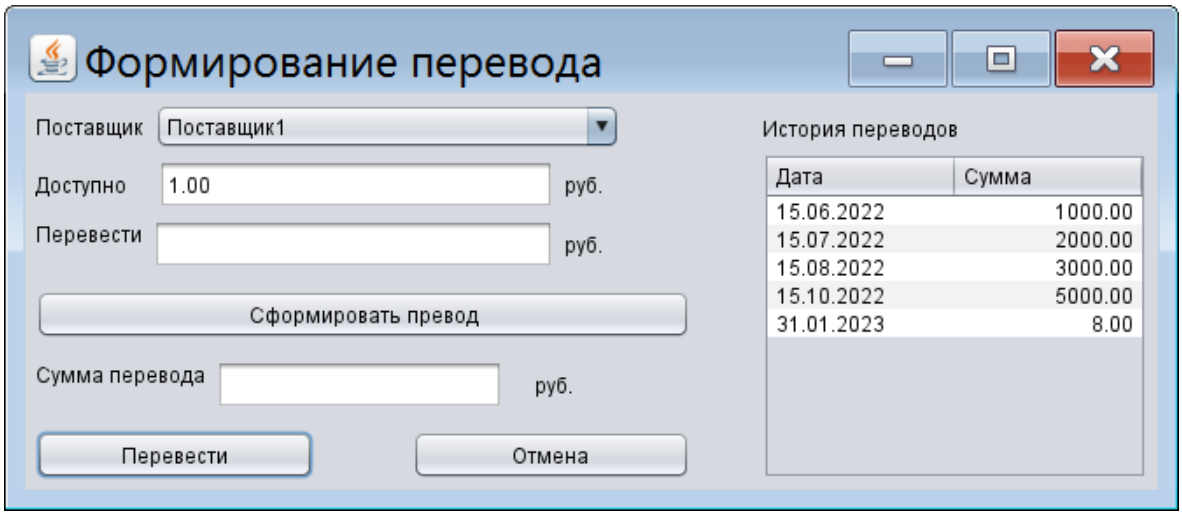

Рисунок 14 – Повторный вызов страницы формирования перевода

После проведения теста база данных импортирована в Excel. Состояние базы данных после проведения теста показано в таблицах 7-10.

Таблица 7 - Результатное состояние таблицы потребителей

Id_Consumer	ConsumerName	ConsumerSet	Supplierid_Supplier
1	Потребитель1	лицевой1	1
2	Потребитель2	лицевой2	2
3	Потребитель3	лицевой3	1
4	НовыйПотребитель	НовыйЛицевой	3

Таблица 8 - Результатное состояние таблицы платежей

Id_Payment	PaymentDate	PaymentSum	Mark	Consumerid _Consumer
1	01.01.2023	10.00	0	1
2	02.01.2023	1.00	1	1
3	03.01.2023	2.00	0	1
4	04.01.2023	7.00	1	1
5	05.01.2023	6.00	0	1
6	06.01.2023	1.00	0	1
7	07.01.2023	5.00	0	1
8	08.01.2023	1.00	1	1
9	09.01.2023	1.00	0	2
10	10.01.2023	1.00	0	3
11	31.01.2023	555.55	0	4

Таблица 9 - Результатное состояние таблицы переводов

Id_Remittance	RemittanceDate	RemittanceSum	Supplierid_supplier
1	15.06.2022	1 000.00	1
2	15.07.2022	2 000.00	1
3	15.08.2022	3 000.00	1
4	15.09.2022	1 000 000.00	2
5	15.10.2022	5 000.00	1
6	31.01.2023	8.00	1

Таблица 10 - Результатное состояние таблицы поставщиков

Id_Supplier	SupplierName	BankName	BankBIK	SupplierSet
1	Поставщик1	Банк1	123456789	расчСчет1
2	Поставщик2	Банк2	123456790	расчСчет2
3	НовыйПоставщик	НовыйБанк	НовыйБИК	НовыйСчет

Изменения в таблицах базы данных соответствуют требованиям теста. Тестирование разработанного программного обеспечения выполнено успешно.

Выводы по разделу:

В третьем разделе выполнен выбор средств разработки, разработано программное обеспечение для автоматизации денежных переводов поставщикам услуг, проведено тестирование разработанного программного обеспечения.

Заключение

Разработанное программное обеспечение позволяет эффективно осуществлять переводы поставщикам услуг со стороны агрегатора платежей ЖКХ.

В результате постановки задачи выполнен обзор предметной области, сформированы требования к программному обеспечению и проведено их ранжирование по значимости, проанализированы аналогичные существующие решения.

Полученные в ходе проектирования архитектурные решения, модели и алгоритмы содержат всю необходимую информацию для разработки приложения. Разработанные концептуальная, логическая и физическая модели базы данных позволяют разработать базу данных приложения в СУБД PostgreSQL.

Также выполнено обоснование выбора двухуровневой клиент-серверной архитектуры, проведено логическое моделирование программного обеспечения, в ходе которого разработаны диаграмма прецедентов и модели данных, разработан алгоритм формирования перевода из платежей.

Разработанный алгоритм формирования переводов из платежей основан на типовой задаче Combination Sum 2 и обеспечивает формирование перевода на ближайшую к целевой сумму без разделения платежа.

Разработанные java приложение и PostgreSQL база данных обеспечивают редактирование справочников, ввод платежей и формирование перевода. По результатам тестирование разработанное приложение работает корректно.

Практическое использование разработанного программного обеспечения позволит эффективно формировать переводы поставщикам услуг из наборов платежей потребителей.

Таким образом, все поставленные задачи были успешно выполнены, цель работы – достигнута.

Список используемой литературы и используемых источников

1. Анализ популярных реляционных систем управления базами данных (2022 г) [Электронный ресурс]. URL: https://drach.pro/blog/hi-tech/item/196-popular-relational-dbms-2022 (дата обращения: 01.02.2023).
2. Архитектура «Клиент-сервер» [Электронный ресурс]. URL: https://itelon.ru/blog/arkhitektura-klient-server/?ysclid=l93uf84y2c790984084. (дата обращения: 29.02.2023).
3. Выбор IDE для Java-разработки. [Электронный ресурс]. URL: https://javarush.com/groups/posts/1642-eclipse-netbeans-ili-intellij-idea-vihbiraem-ide-dlja-java-razrabotki (дата обращения: 01.02.2023).
4. Заяц А. М., Васильев Н.П. Проектирование и разработка веб-приложений. Введение в frontend и backend разработку на JavaScript и node.js. М.: Издательство «Лань», 2020. 120 с.
5. Интернет-программирование : учеб. пособие / М.А. Колотилина. [Электронный ресурс]. URL: https://lms2.sseu.ru/pluginfile.php/331799/mod_resource/content/2/Колотилина.pdf. (дата обращения: 29.08.2022).
6. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. URL: https://habr.com/ru/post/566218/ (дата обращения: 24.02.2023).
7. Казиев, В. М. Введение в анализ, синтез и моделирование систем: учебное пособие. Москва, Саратов: ИнтернетУниверситет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. 270 с.
8. Классификация методов и моделей эффективности ИС [Электронный ресурс]. URL: https://sites.google.com/site/isefficiency/klassifikacia-izvestnyh-metodov-i-modelej. (дата обращения: 19.02.2023).
9. Язык программирования Java: особенности, синтаксис и идеи для первых проектов [Электронный ресурс]. URL:

https://ru.hexlet.io/blog/posts/yazyk-programirovaniya-java-osobennosti-sintaksis-i-idei-dlya-pervyh-proektov (дата обращения: 01.02.2023).

10. Chen, Peter P.. “The entity-relationship model—toward a unified view of data.” ACM Trans. Database Syst. 1 (2011): 9-36.

11. Chen, Peter. (2002). Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned. 10.1007/978-3-642-59412-0_17.

12. Combination Sum II [Электронный ресурс]. URL: https://leetcode.com/problems/combination-sum-ii/ (дата обращения: 29.01.2023).

13. Connect to PostgreSQL Data in NetBeans [Электронный ресурс]. URL: https://www.cdata.com/kb/tech/postgresql-jdbc-netbeans.rst (дата обращения: 31.01.2023).

14. Java и базы данных [Электронный ресурс]. URL: https://www.osp.ru/news/articles/1997/0511/13032031 (дата обращения: 30.01.2023).

15. LeetCode - Combination Sum II [Электронный ресурс]. URL: https://dev.to/_alkesh26/leetcode-combination-sum-ii-1499 (дата обращения: 29.01.2023).

16. Limiting postgreSQL update command [Электронный ресурс]. URL: https://stackoverflow.com/questions/11432155/limiting-postgresql-update-command (дата обращения: 30.01.2023).

17. NetBeans IDE [Электронный ресурс]. URL: https://ru.bmstu.wiki/NetBeans_IDE (дата обращения: 30.01.2023).

18. PgAdmin PostgreSQL Tools [Электронный ресурс]. URL: https://www.pgadmin.org/ (дата обращения: 31.01.2023).

19. Using Java Netbeans IDE Postgres database [Электронный ресурс]. URL: https://quantlabs.net/blog/2015/03/using-java-netbeans-ide-postgres-database (дата обращения: 31.01.2023).

20. Visual Paradigm [Электронный ресурс]. URL: https://www.visual-paradigm.com/ (дата обращения: 31.01.2023).