Федеральное государственное автономное образовательное

учреждение высшего профессионального образования «НИТУ МИСиС»

Институт «Информационных технологий и компьютерных наук»

Кафедра «Автоматизированных систем управления»

Курсовая работа

по курсу «Технологии программирования»

Тема:

«Разработка клиент-серверного приложения с использованием базы данных»

Выполнила: студентка 2 курса,

группы БИВТ-21-8 Порошина М. А.

Руководитель: Козлов М. Е.

 Москва 2022

**Оглавление**

[1. Введение 3](#_Toc124473235)

[2. Теоретическая часть 4](#_Toc124473236)

[2.1. Язык программирования Java 4](#_Toc124473237)

[2.2. Архитектурный паттерн «Клиент-сервер» 4](#_Toc124473238)

[2.3. Архитектурный паттерн «MVC» (Model-View-Controller, Модель-Представление-Контроллер) 4](#_Toc124473239)

[2.4. Spring Framework 4](#_Toc124473240)

[2.5. Библиотека Hibernate 5](#_Toc124473241)

[2.6. База данных PostgreSQL 5](#_Toc124473242)

[2.7. Формат CSV 5](#_Toc124473243)

[3. Практическая часть 6](#_Toc124473244)

[3.1. Детальные требования к программе 6](#_Toc124473245)

[3.2. Декомпозиция на задачи 6](#_Toc124473246)

[3.3. Разработка 7](#_Toc124473247)

[3.3.1. Подключение к СУБД PostgreSQL и создание БД 7](#_Toc124473248)

[3.3.2. Создание моделей таблиц и репозиториев к ним 8](#_Toc124473249)

[3.3.3. Создание сервиса для чтения CSV-файлов и сохранения полученных данных в БД 9](#_Toc124473250)

[3.3.4. Создание сервиса, реализующего SQL-запросы к БД 10](#_Toc124473251)

[3.3.5. Создание контроллеров для работы с веб-страницами и самих веб-страниц 11](#_Toc124473252)

[3.4. Тестирование 13](#_Toc124473253)

[4. Заключение 15](#_Toc124473254)

[5. Список литературы 16](#_Toc124473255)

# **1. Введение**

В настоящее время существует множество приложений, позволяющих их пользователям работать с большими объемами данных. Среди них можно выделить клиент-серверные приложения, работа которых непосредственно связана с взаимодействием с базами данных при помощи языка программирования SQL и SQL-запросов.

Постановка задачи:

Целью курсовой работы является разработка клиент-серверного приложения с использованием базы данных.

Приложение должно включать следующий функционал:

* Работа с данными в CSV-файлах;
* Автоматическое создание таблиц в базе данных;
* Веб-страница для загрузки данных в таблицы базы данных из CSV-файлов;
* Вывод результатов в CSV-файлы;
* Выделение дня транзакции в отдельную колонку в таблице транзакций;
* Веб-страница вывода средних значений по полю amount, сгруппировав данные по дню, а в рамках дня по MCC коду со следующими условиями: MCC коды, встречаются в выборке более чем 60000 раз.

# **2. Теоретическая часть**

## **2.1. Язык программирования Java**

Язык программирования Java – строго типизированный объектно-ориентированный язык программирования общего назначения.

Приложения Java транслируются в специальный байт-код, ввиду чего могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины.

## **2.2. Архитектурный паттерн «Клиент-сервер»**

В архитектуре «Клиент-сервер» функциональность системы обеспечивается набором услуг (сервисов). Каждый сервис располагается на своем сервере. Клиенты являются пользователями этих сервисов, для чего обращается к серверу. В роли клиентов выступают клиентские программы, часто запускаемые одновременно на различных компьютерах.

Взаимодействие клиента и сервера осуществляется при помощи сети. Это обусловлено тем, что клиент-серверные системы реализованы как распределенные системы, соединенные с помощью интернет-протокола.

## **2.3. Архитектурный паттерн «MVC» (Model-View-Controller, Модель-Представление-Контроллер)**

Паттерн «MVC» отделяет представление системы и взаимодействие с системой от данных системы. Система разделяется на три логических компонента, которые взаимодействуют друг с другом.

Компонент Model (Модель) управляет системными данными и операциями над данными. Компонент View (Представление) отображает данные для пользователя. Компонент Controller (Контроллер) взаимодействует с пользователем, инициирует операции в модели и управляет работой представления.

## **2.4. Spring Framework**

Spring Framework — это универсальный фреймворк с открытым исходным кодом для Java-платформы. Особенность данного фреймворка заключается в том, что он сам создает объекты классов и вызывает методы, написанные пользователем.

## **2.5. Библиотека Hibernate**

Hibernate — это Java библиотека с открытым исходным кодом, представляющая собой инструмент объектно-реляционного отображения (object-relational mapping — ORM). Его основной задачей является преобразование данных реляционной БД в объектно-ориентированные модели и обратно. Фреймворк также предоставляет средства для автоматического построения запросов, поиска и извлечения данных.

## **2.6. База данных PostgreSQL**

PostgreSQL – это свободная объектно-реляционная система управления базами данных (СУБД). Данная СУБД базируется на языке SQL.

Среди преимуществ пользования PostgreSQL можно выделить:

* Высокопроизводительные и надежные механизмы транзакций и репликации;
* Расширяемую систему встроенных языков программирования;
* Наследование;
* Встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексикации;
* Расширяемость (возможность создания новых типов данных, типов индексов, языков программирования, подключение любых внешних источников).

## **2.7. Формат CSV**

Формат CSV (comma-separated values) – текстовый формат, предназначенный для представления табличных данных. Строки таблиц соответствуют строкам текста, которые содержат одно или несколько полей, разделенных запятыми, точками с запятой или знаками табуляции.

# **3. Практическая часть**

## **3.1. Детальные требования к программе**

По заданию клиент-серверное приложение должно иметь следующий функционал:

* Работа с данными в CSV-файлах и их загрузка в БД;
* Автоматическое создание таблиц;
* Веб-страница для загрузки CSV-файлов;
* Веб-страницы для вывода результатов реализуемых функций;
* Вывод результатов реализуемых функций в CSV-файлы;
* Реализация функции выделение дня транзакции в отдельную колонку в таблице транзакций;
* Реализация функции вывода средних значений по полю amount, сгруппировав данные по дню, а в рамках дня по MCC коду со следующими условиями: MCC коды, встречаются в выборке более чем 60000 раз.

## **3.2. Декомпозиция на задачи**

На основе детальных требований выделю задачи, которые реализую в процессе работы:

1. Подключиться к СУБД PostgreSQL и создать БД;
2. Создать моделей для автоматического создания таблиц БД:
   1. GenderTrain;
   2. Transactions;
   3. TrMccCodes;
   4. TrTypes.
3. Создать репозитории для каждой модели:
   1. GenderTrainRepository;
   2. TransactionsRepository;
   3. TrMccCodesRepository;
   4. TrTypesRepository.
4. Создать сервис для чтения CSV-файлов и сохранения полученных данных в БД CsvParserService;
5. Создать сервис, реализующий SQL-запросы к БД в соответствии с заданием и сохранением полученных результатов в CSV-файлах FunctionsService;
6. Создать контроллеры для работы с веб-страницами:
   1. CsvParserController для работы с CSV-файлами;
   2. FunctionController для работы с функциями.
7. Создать веб-страницы:
   1. Для загрузки файлов;
   2. Для выбора функций;
   3. Для вывода результатов каждой из функций.

## **3.3. Разработка**

### **3.3.1. Подключение к СУБД PostgreSQL и создание БД**

В приложении pgAdmin создам базу данных bank (рис. 1).

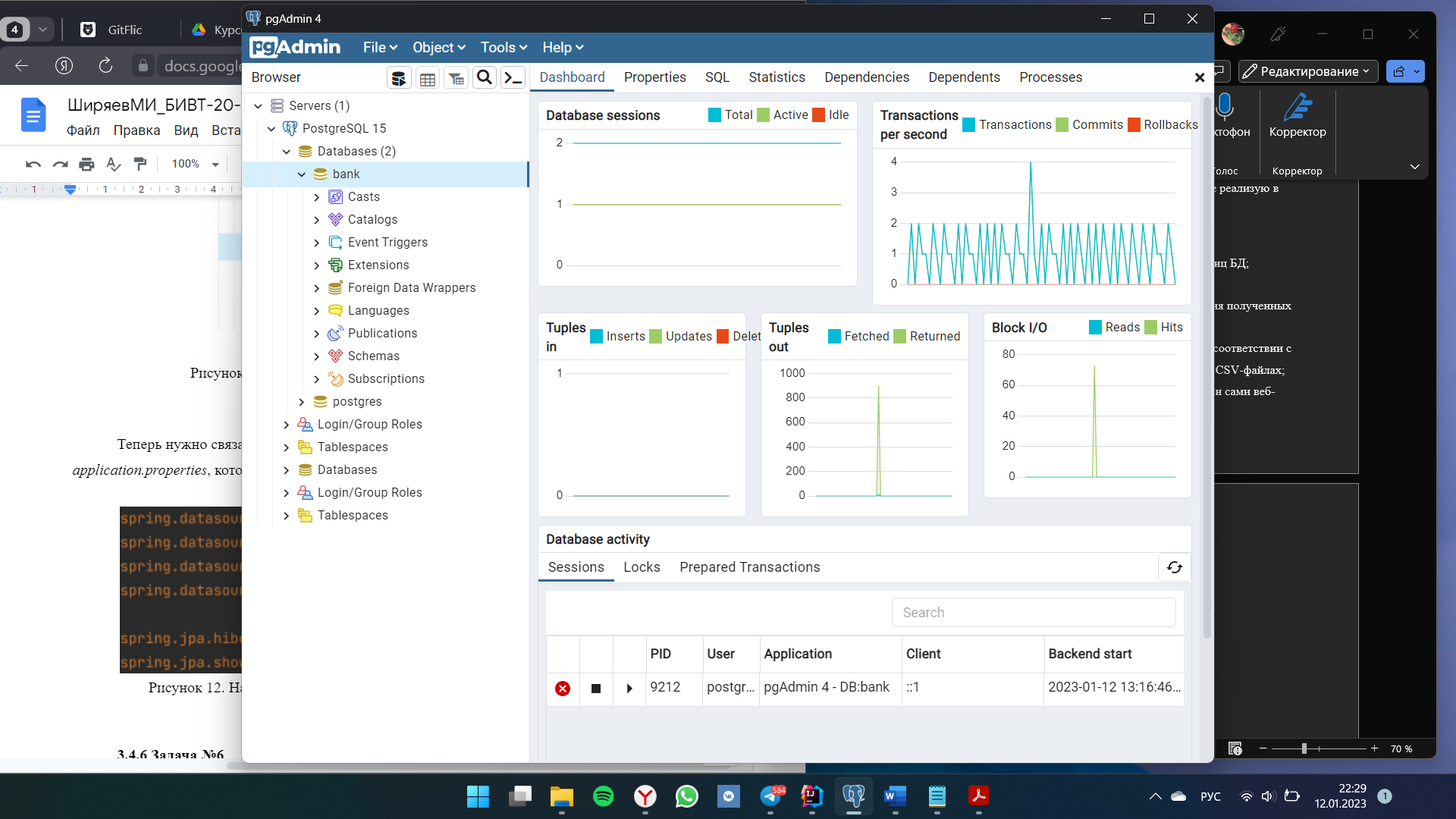


Рисунок 1. Создание БД bank в pgAdmin

Привяжу БД к приложению. Для этого указываю атрибуты подключения в файле application.properties, что показано на рисунке 2.

Изображение выглядит как текст, монитор, электроника, снимок экрана

Автоматически созданное описание

Рисунок 2. Подключение к БД

### **3.3.2. Создание моделей таблиц и репозиториев к ним**

Создам модели таблиц GenderTrain, Transactions, TrMccCodes, TrTypes БД bank. Рассмотрим модель таблицы на примере GenderTrain, продемонстрированном на рисунке 3. Каждая модель имеет аннотации @Entity и @Table, которые показывают, что данная сущность является таблицей БД. Аннотация @Id закрепляет атрибут, являющийся её первичным ключом.

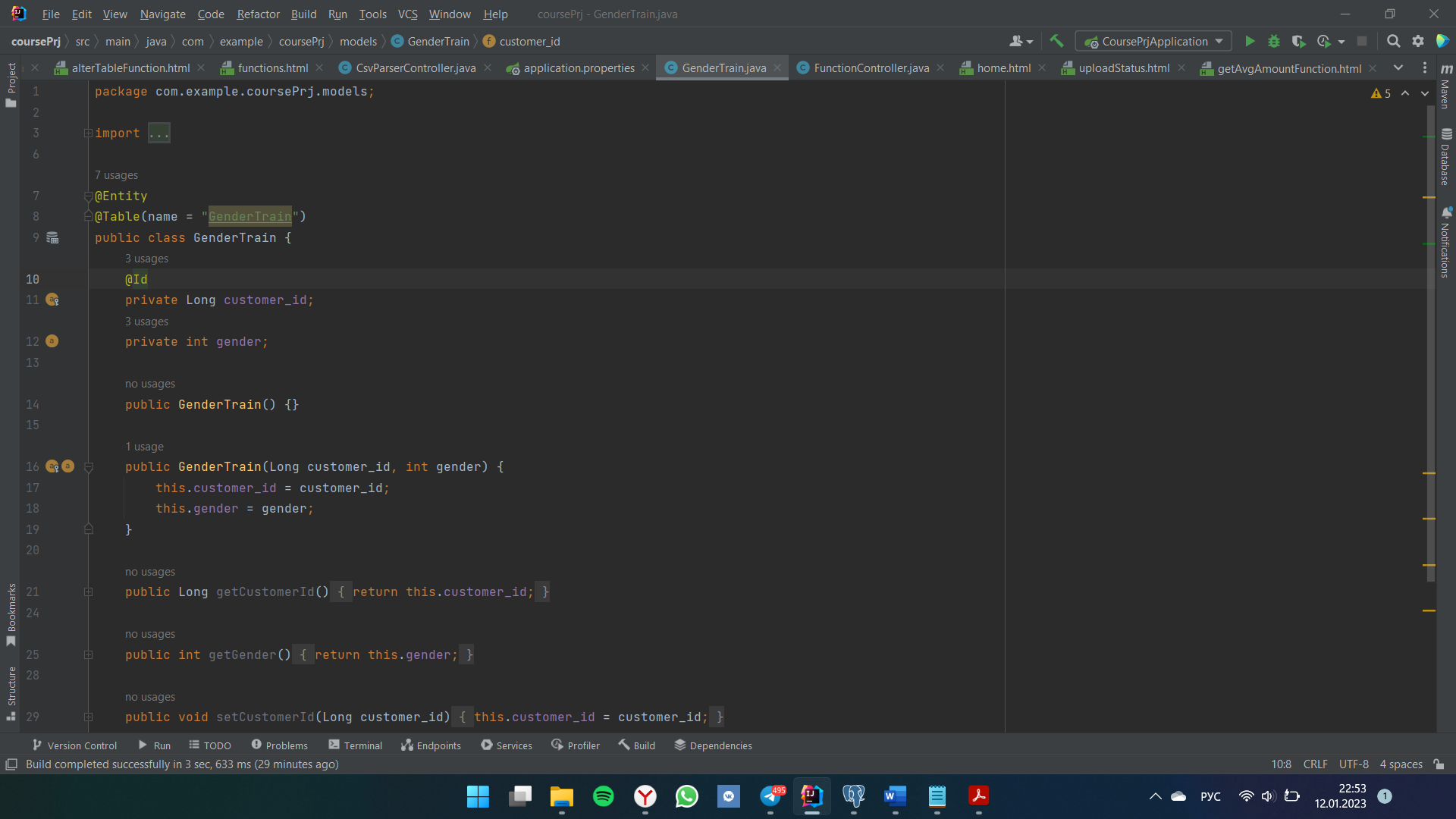


Рисунок 3. Создание модели таблицы GenderTrain

Для каждой модели создам репозиторий: GenderTrainRepository, TransactionsRepository, TrMccCodesRepository, TrTypesRepository. На примере GenderTrainRepository, продемонстрированном на рисунке 4, видим, что репозитории реализованы в виде интерфейсов, наследующих интерфейс JpaRepository, для работы с представлениями таблиц.

Изображение выглядит как текст, монитор, электроника, снимок экрана

Автоматически созданное описание

Рисунок 4. Создание репозитория модели GenderTrain

### **3.3.3. Создание сервиса для чтения CSV-файлов и сохранения полученных данных в БД**

Создам сервис CsvParserService для чтения данных из CSV-файлов в таблицв БД. На рисунке 5 видно, что сервис помечен аннотацией @Service, а используемые репозитории - аннотацией @Autowired.

Изображение выглядит как текст, монитор, снимок экрана, компьютер

Автоматически созданное описание

Рисунок 5. Создание сервиса CsvParserService

Реализация работы сервиса заключается в передаче прочитанных данных методом upload (рис. 5) в класс для сохранения в таблицу БД на основе названия самого файла. Пример класса для записи данных показан на рисунке 6.

Изображение выглядит как текст, монитор, снимок экрана, электроника

Автоматически созданное описание

Рисунок 6. Метод добавления прочитанных данных в БД

### **3.3.4. Создание сервиса, реализующего SQL-запросы к БД**

Создам сервис FunctionsService для работы с SQL-запросами к БД. Данный сервис содержит два метода – divideColumn и getAvgAmount.

Метод divideColumn, показанный на рисунке 7, выполняет SQL-запрос, реализующий функцию выделения дня транзакции в отдельную колонку в таблице транзакций и сохранение всех данных таблицы Transactions в CSV-файл, получаемых также с помощью SQL-запроса.

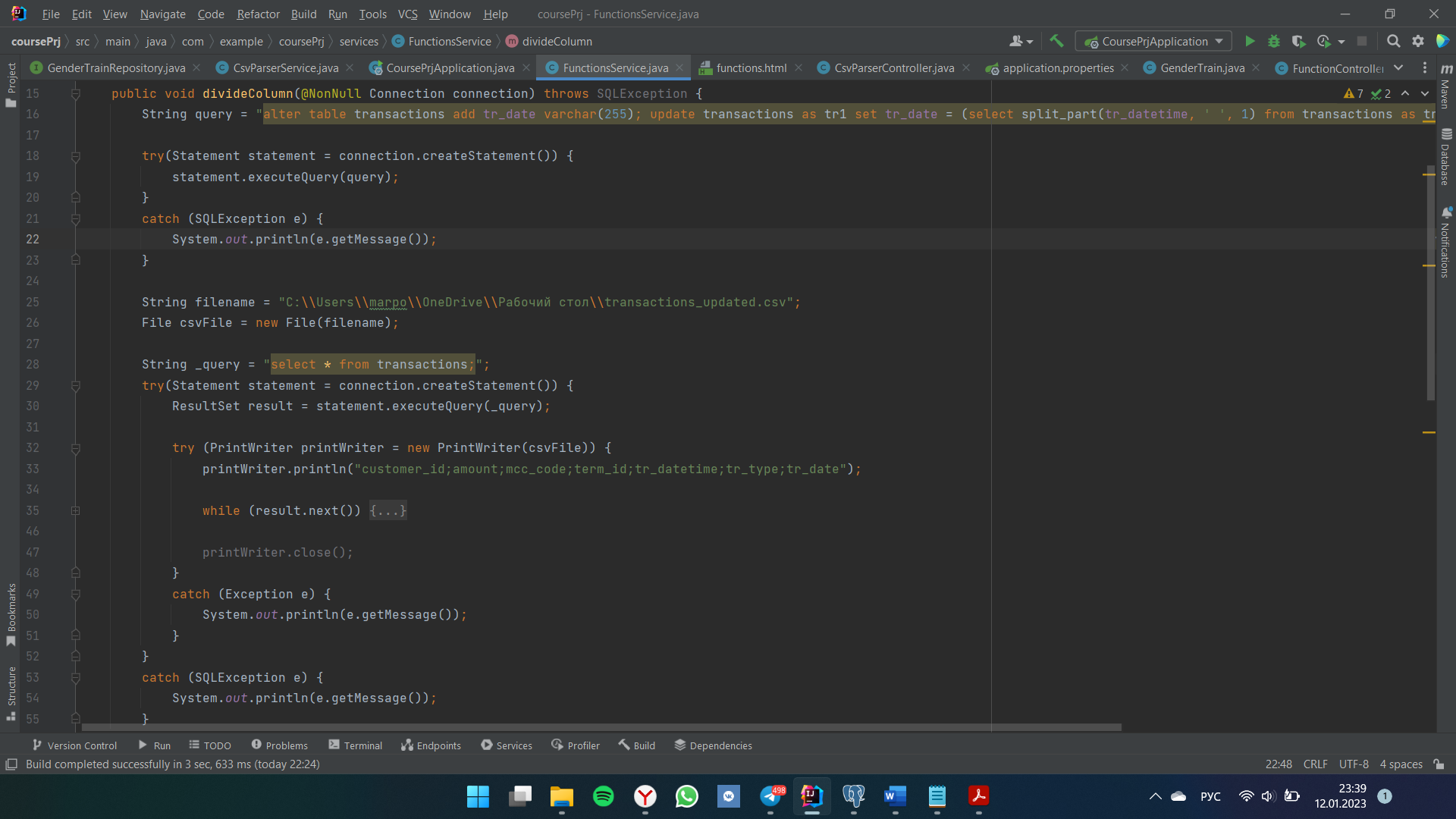


Рисунок 7. Реализация функции divideColumn

Метод getAvgAmount, показанный на рисунке 8, выполняет SQL-запрос, реализующий функцию вывода средних значений по полю amount, сгруппировав данные по дню, а в рамках дня по MCC коду, а также сохранение полученного столбца средних значений в CSV-файл.

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Рисунок 8. Реализация функции getAvgAmount

### **3.3.5. Создание контроллеров для работы с веб-страницами и самих веб-страниц**

В проекте реализовано два контроллера: CsvParserController для работы с CSV-файлами и FunctionController для работы с функциями. Контроллеры помечены аннотацией @Controller, а методы имеют аннотации @GetMapping или @PostMapping для обработки GET-запросов или POST-запросов соответственно.

CsvParserController (рис. 9) обрабатывает запросы, связанные с переходом на главную страницу и с работой с CSV-файлами, и передает массив файлов сервису CsvParserService.

Изображение выглядит как текст, снимок экрана, монитор, электроника

Автоматически созданное описание

Рисунок 9. Реализация контроллера CsvParserController

FunctionController (рис. 10) обрабатывает запросы, связанные с переходом на страницы реализации функций и их выполнением. В методах контроллера реализуется подключение к БД и выполнение функций.

Изображение выглядит как текст, монитор, снимок экрана, электроника

Автоматически созданное описание

Рисунок 10. Реализация контроллера FunctionController

Созданные в проекте веб-страницы показаны на рисунке 11.

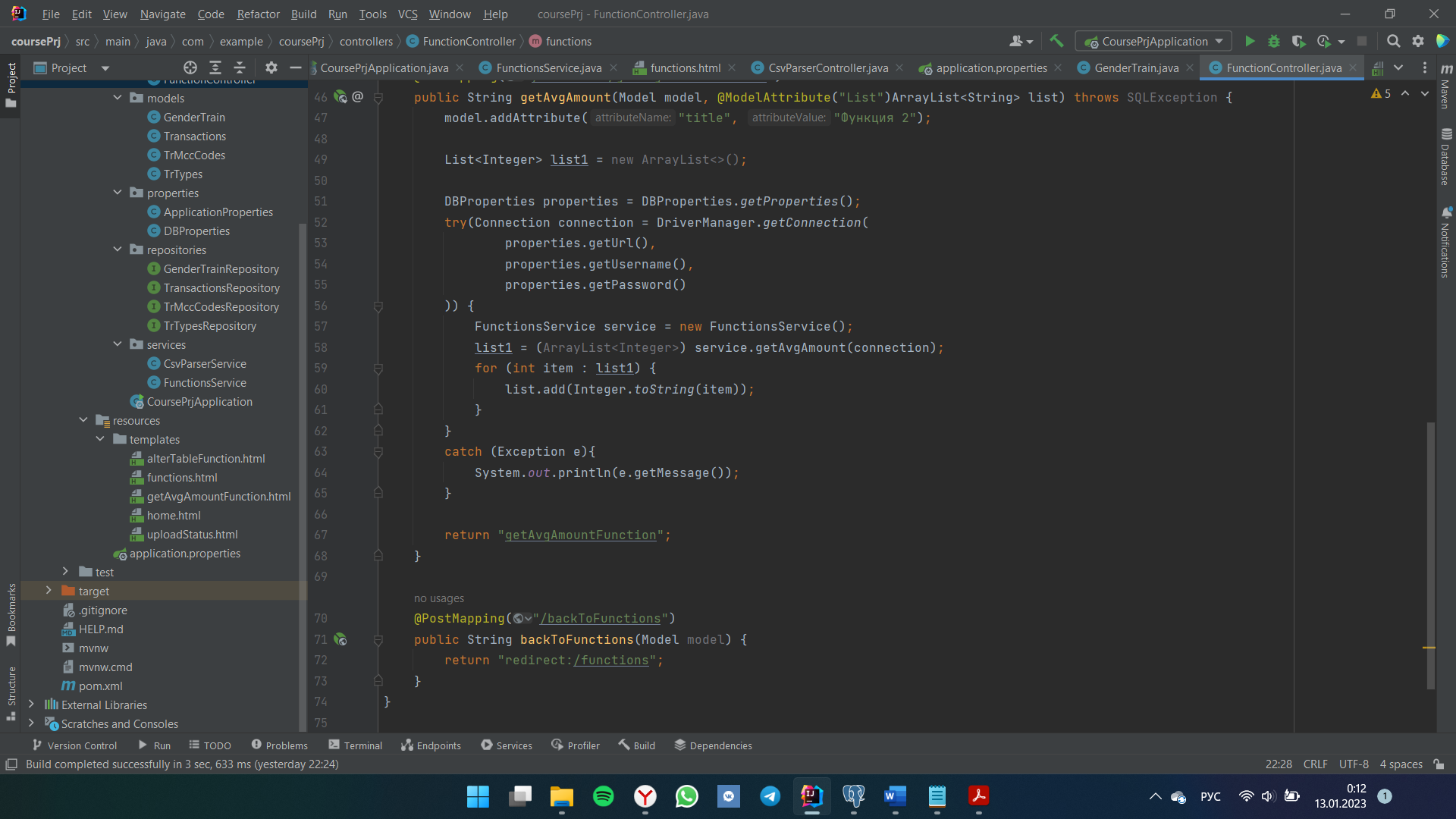


Рисунок 11. Созданные веб-страницы

## **3.4. Тестирование**

Запущу созданное приложение. Как видно на рисунке 13, модели таблиц были успешно созданы.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13. Создание таблиц в БД

Для тестирования работы приложения я создала CSV-файлы и заполнила их данными для добавления в БД. Загружу их через веб-страницу (рис. 14).

Изображение выглядит как текст, снимок экрана, монитор

Автоматически созданное описание

Рисунок 14. Загрузка тестовых файлов

На рисунке 15 показано, что данные из файлов были успешно добавлены в таблицы БД.

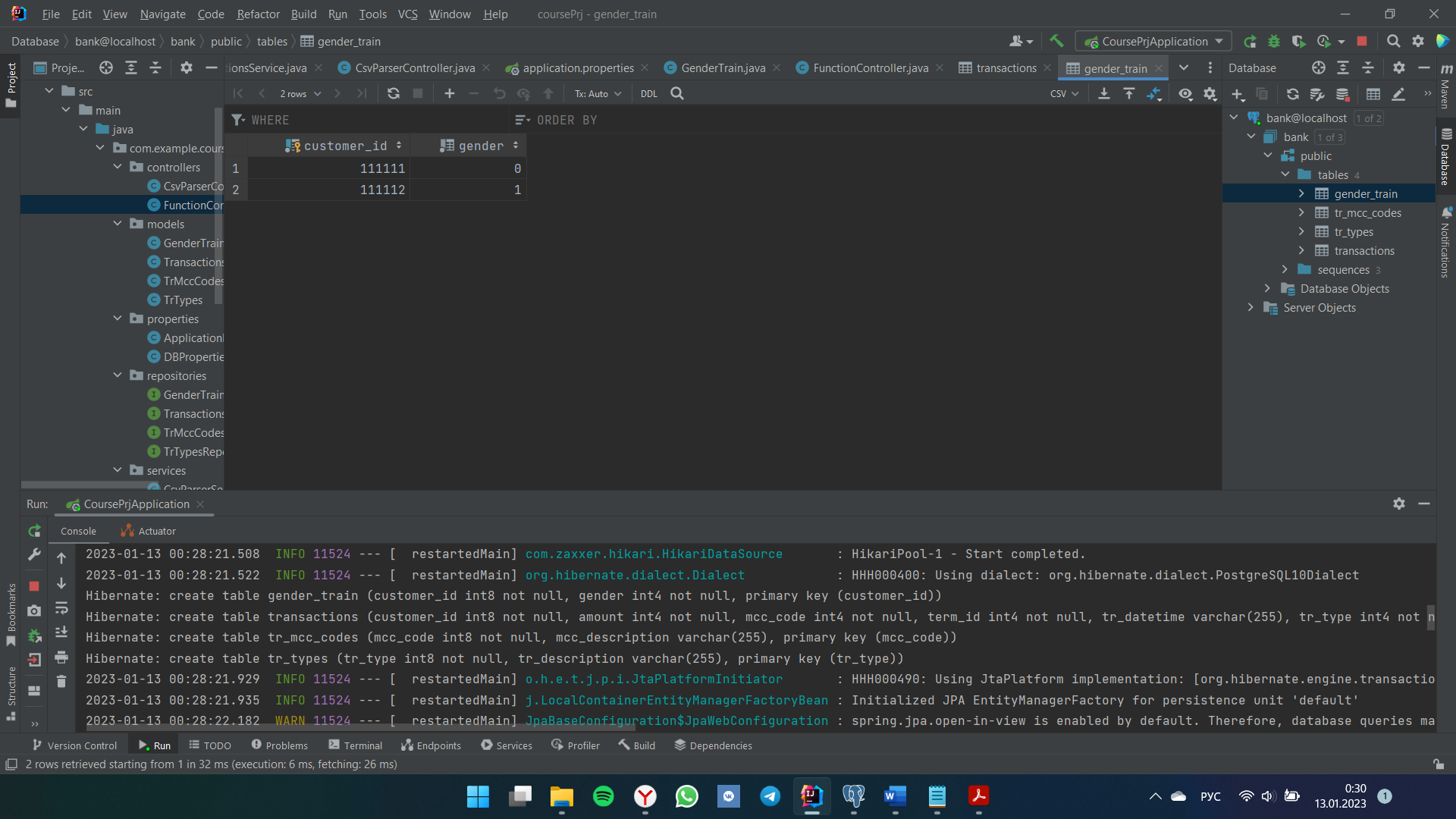


Рисунок 15. Успешное заполнение таблиц

Поочередно выполню обе функции. На рисунке 17 виден результат выполнения функции 1, а на рисунках 16 и 18 – результат выполнение функции 2.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16. Результат выполнения функции 2.

Изображение выглядит как текст, монитор, компьютер, внутренний

Автоматически созданное описание

Рисунок 17. Файл с результатом функции 1

Изображение выглядит как текст, монитор, внутренний, компьютер

Автоматически созданное описание

Рисунок 18. Файл с результатом функции 2

# **4. Заключение**

В процессе выполнения данной курсовой работы было разработано клиент-серверное приложение с использованием базы данных. В рамках работы над проектом были реализованы следующие задачи:

* Работа с данными в CSV-файлах;
* Автоматическое создание таблиц в базе данных;
* Веб-страницы для загрузки CSV-файлов, выводов результатов реализованных функций;
* Вывод результатов в CSV-файлы;
* Реализация функций на основе SQL-запросов к базам данных.

В процессе разработки данного приложения были повторены навыки работы с базами данных при помощи СУБД PostgreSQL, получение и запись данных в CSV-файлы, а также изучена работа с Spring Framework, создание и вывод данных на веб-страницы.

В ходе тестирования была подтверждена корректная работа приложения.

# **5. Список литературы**

1. Орлов C. Программная инженерия. Учебник для вузов. 5-е издание обновленное и дополненное. Стандарт третьего поколения. — СПб.: Питер, 2016. — 640 с.:
2. К. Сьерра «Изучаем Java» - Эксмо, 2012–720 с.
3. Документация Spring Framework [электронный ресурс]: <https://spring.io/>
4. Документация PostgreSQL [электронный ресурс]: <https://postgrespro.ru/docs/postgresql>
5. Документация Thymeleaf [электронный ресурс]: https:/[/www](http://www.thymeleaf.org/documentation.html).[thymeleaf.org/documentation.html](http://www.thymeleaf.org/documentation.html)
6. Работа с CSV-файлами [электронный ресурс]: https://www.baeldung.com/java-csv-file-array