МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Казанский национальный исследовательский технический

университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

*(наименование института (факультета)*

Кафедра динамики процессов и управления

*(наименование кафедры)*

Лабораторная работа № 4

По дисциплине «Программирование»

|  |  |
| --- | --- |
| Выполнил: | обучающийся группы 4165\_ |
|  | Шестаков Михаил |

Казань 2025 г.

**Задание №1**

**Цель:** импорт CSV → ADO.NET → экспорт в JSON/XML.

**Требуется:**

1. Реализовать парсер books.csv (формат: ISBN, Title, Author, Year, Pages). Учесть кавычки и запятые внутри полей.
2. Через ADO.NET вставить записи в таблицу Books. Использовать параметризованные запросы и обработку ошибок.
3. Сохранить весь список в books.json и books.xml (полная сериализация всех полей).
4. Записать лог с количеством успешно вставленных записей и перечнем ошибок в файл логов.

**Объяснение работы программы**

В рамках задачи был реализован импорт данных с csv файла в базу данных PostgreSQL с использованием парсера данных, корректно учитывающий стандартные особенности формата, включая экранирование кавычек и обработку полей, содержащих запятые. Метод ParseCsv считывает файл построчно (с пропуском заголовка), разбирает каждую строку с помощью вспомогательного метода ParseCsvLine, и создаёт объекты типа Book. При обнаружении некорректного формата (неверное количество полей или ошибки преобразования типов) выбрасываются исключения с указанием номера строки и контекста ошибки, что обеспечивает удобную диагностику проблем с исходными данными.

Загруженные данные последовательно вставляются в таблицу Books с использованием ADO.NET и драйвера Npgsql. Для обеспечения целостности данных вся операция выполняется в рамках одной транзакции: в случае возникновения ошибки транзакция откатывается, а информация об ошибке записывается в лог (что учитывает выполнение 4 задания). Вставка реализована с применением параметризованных SQL-запросов, что исключает риски SQL-инъекций и гарантирует корректную обработку специальных символов в данных. Все логи о выполнении показываются в консоли и сохраняются в файл.

**Задание №2**

**Цель:** перейти к EF Core и реализовать CRUD + синхронизацию.

**Требуется:**

1. Создать EF Core модель Book и BooksContext. Добавить миграции и применить их.
2. Реализовать CRUD-операции (консольное меню или аргументы командной строки): добавить книгу, обновить поля, удалить по ISBN, найти по автору/части названия.
3. Реализовать команду "sync" — сравнить записи в БД и в books.json, вывести список различий: несоответствующие записи по ISBN, отсутствующие в одной из сторон.
4. Реализовать также следующий возможности регулирования конфликтов: skip / update\_db / update\_file.

**Объяснение работы программы**

В рамках этой задачи была добавлена реализация системы управления каталогом книг на основе EF Core. Создан класс контекста BooksContext, настроенный для работы с PostgreSQL, через провайдера Npgsql. Для хранения данных определена модель сущности Book, в которой ISBN используется в качестве первичного ключа. Была добавлена и применена миграция, обеспечивающая корректное создание таблицы Books в базе данных.

Реализован набор CRUD-операций – удаление, создание, обновление и поиск. Все операции сопровождаются обработкой исключений и логированием через службу SimpleLogger (что учитывает выполнение 4 задания).

Также реализована команда sync, обеспечивающая двустороннюю синхронизацию между базой данных и внешним JSON-файлом (books.json). Служба SyncService сравнивает записи по ISBN и выявляет три типа расхождений:

* записи, присутствующие только в БД;
* записи, присутствующие только в JSON-файле;
* конфликтующие записи с различающимися данными.

Поддерживается несколько стратегий разрешения расхождений:

* update\_db — обновление базы данных на основе JSON;
* update\_file — перезапись JSON-файла актуальными данными из БД;
* skip — пропуск всех различий без изменений.

При этом обновление JSON-файла выполняется программно: все книги из базы сериализуются в формат JSON с сохранением читаемой структуры.

**Задание №4**

**Цель:** обеспечить транзакционную надёжность и покрытие тестами.

**Требуется:**

1. Реализовать транзакционный импорт так, чтобы при неудаче экспортных/валидационных шагов изменения откатывались.
2. Логгировать успешные и неуспешные транзакции

**Объяснение работы программы**

Во всем проекте реализовано логирование всех важных моментов, всех возможных ошибок. Также реализован транзакционный импорт так, чтобы изменения откатывались при неудаче