**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ бюджетное образовательное учреждение высшего профессионального образования**

**«Санкт-Петербургский государственный электротехнический университет**

**“ЛЭТИ” им.В.И.Ульянова (Ленина)»**

**КАФЕДРА МОЭВМ**

**ОТЧЕТ**

**по лабораторно-практической работе № 7**

**«Построение отчётов в PDF- и HTML- форматах»**

**по дисциплине «Объектно - ориентированное программирование на языке Java»**

Выполнил: Шарапов И.Д.

Факультет: КТИ

Группа: №3312

Подпись преподавателя: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**Содержание**

[Цель работы 3](#_Toc182753868)

[Построение шаблона 3](#_Toc182753869)

[Содержимое файлов 7](#_Toc182753870)

[Текст программы 7](#_Toc182753871)

[Приложение 15](#_Toc182753872)

# Цель работы

Знакомство со способами формирования отчётов с использованием конструктора Jaspersoft Studio.

# Построение шаблона

Создадим новый отчёт, для этого в Jaspersoft Studio нажмём File New Jasper Report. Далее выберем шаблон (Рис.1) и нажмём кнопку Next. Далее выберем имя и место сохранения нашего шаблона (Рис. 2) нажмём кнопку Next.

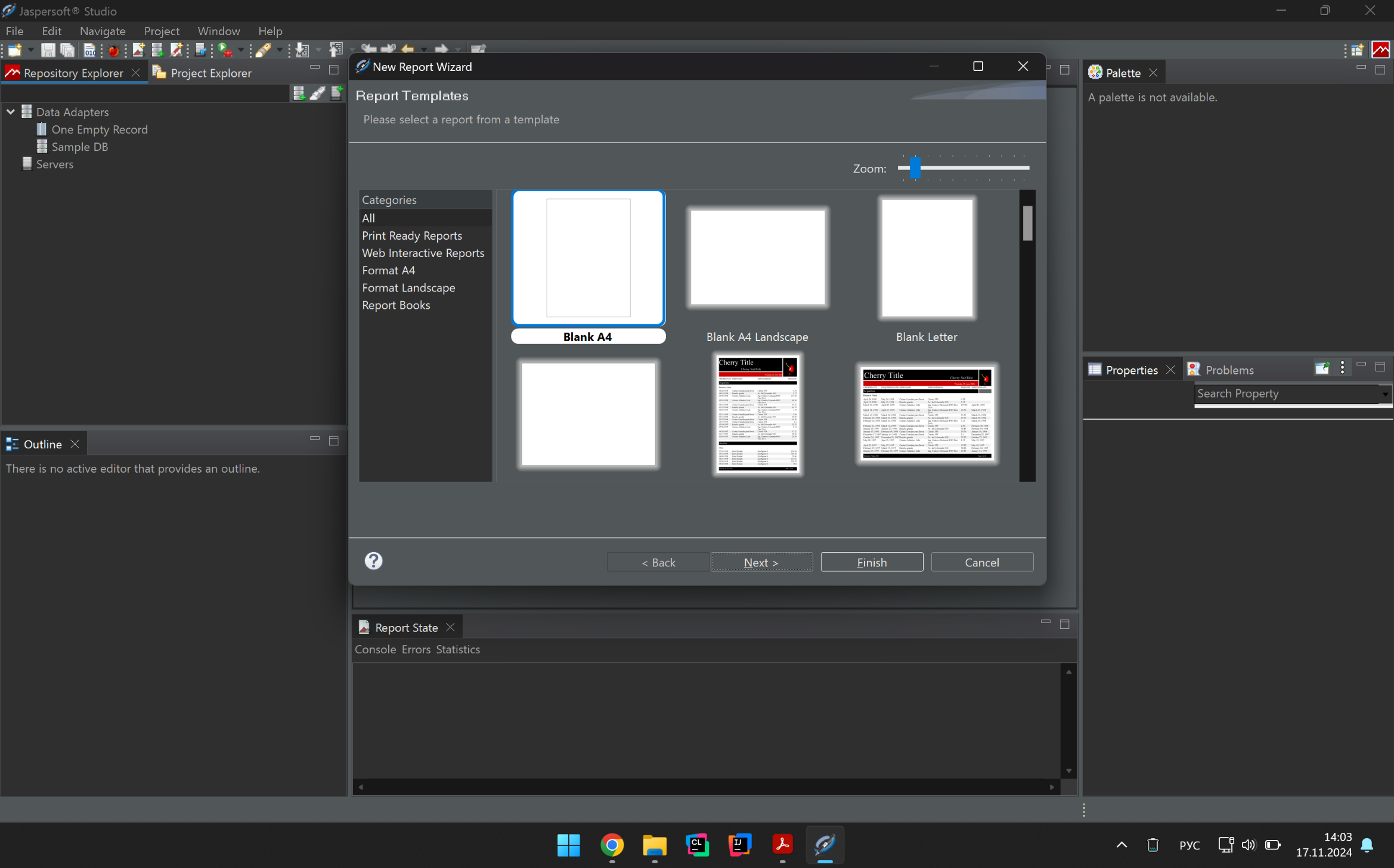


Рисунок 1 – Выбор шаблона

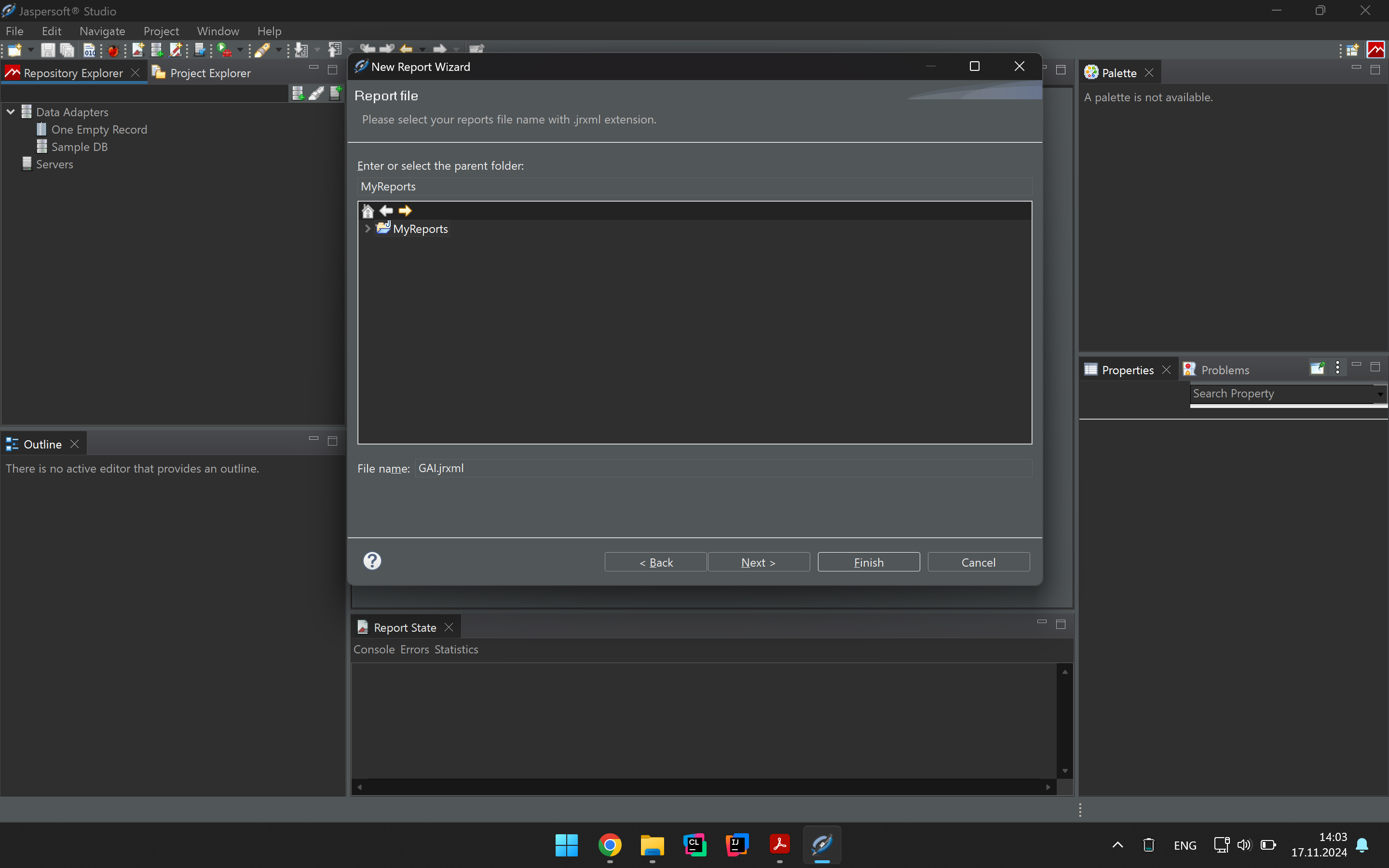


Рисунок 2 – Выбор папки для файла

Открылось диалоговое окно с выбором источника данных (Рис. 3). Введём путь к нашему файлу XML и в графе Select Expression введём «/drivers/driver». Нажмём кнопку Finish.

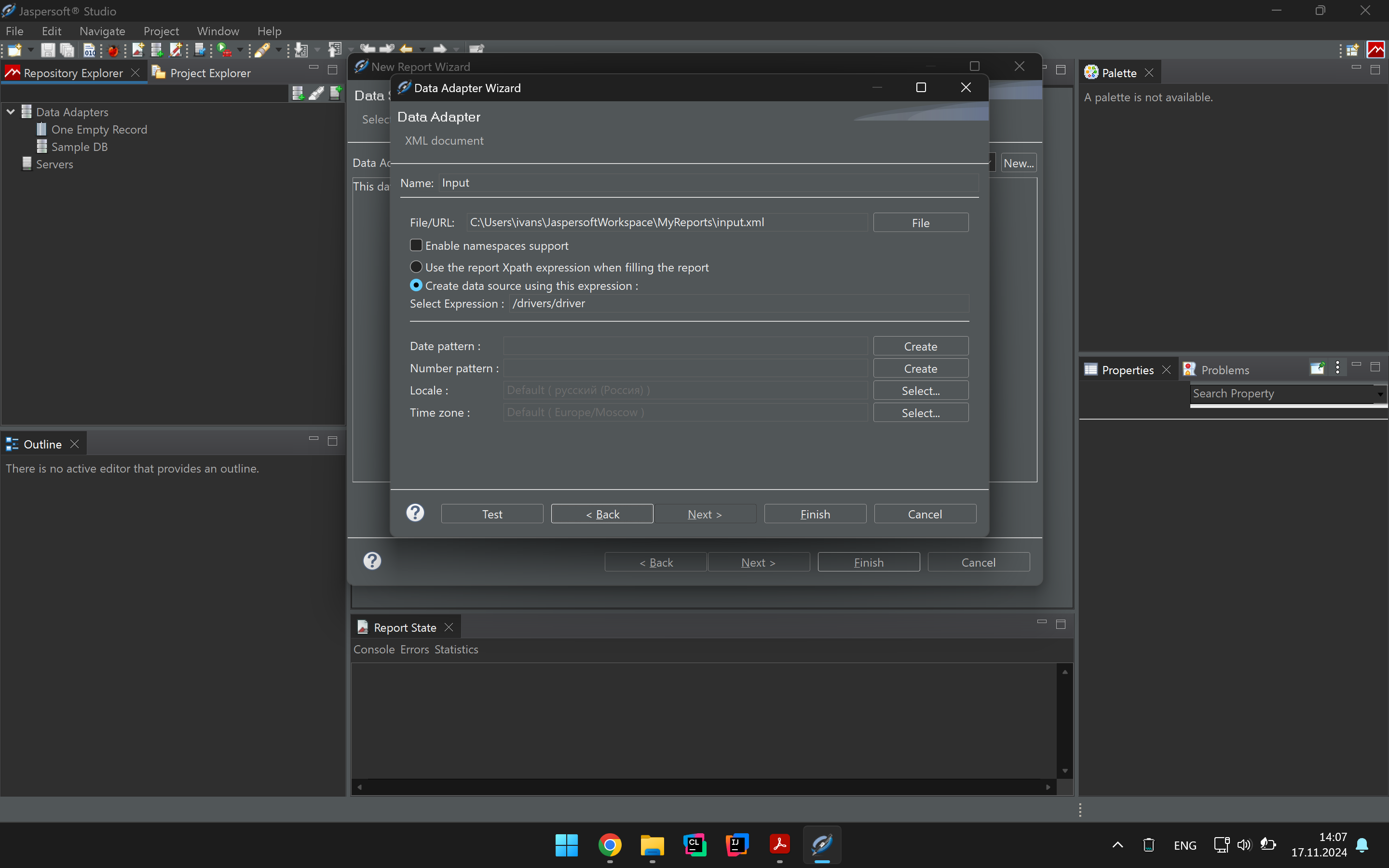


Рисунок 3 – Выбор данных

Теперь перед нами основное окно конструктора (Рис. 4). Здесь мы можем собрать из предложенных элементов скелет нашего отчёта. Создадим подходящий для нашей структуры XML (Рис. 5). Для этого из блока Outline Fields перетащим в Detail 1 нужные поля. Также украсим наш отчёт элементами из Palette. Нажмём на вкладку Preview и увидим наш отчёт (Рис. 6). Здесь же можем экспортировать наш отчёт в множество форматов (PDF, HTML, и многие др.).

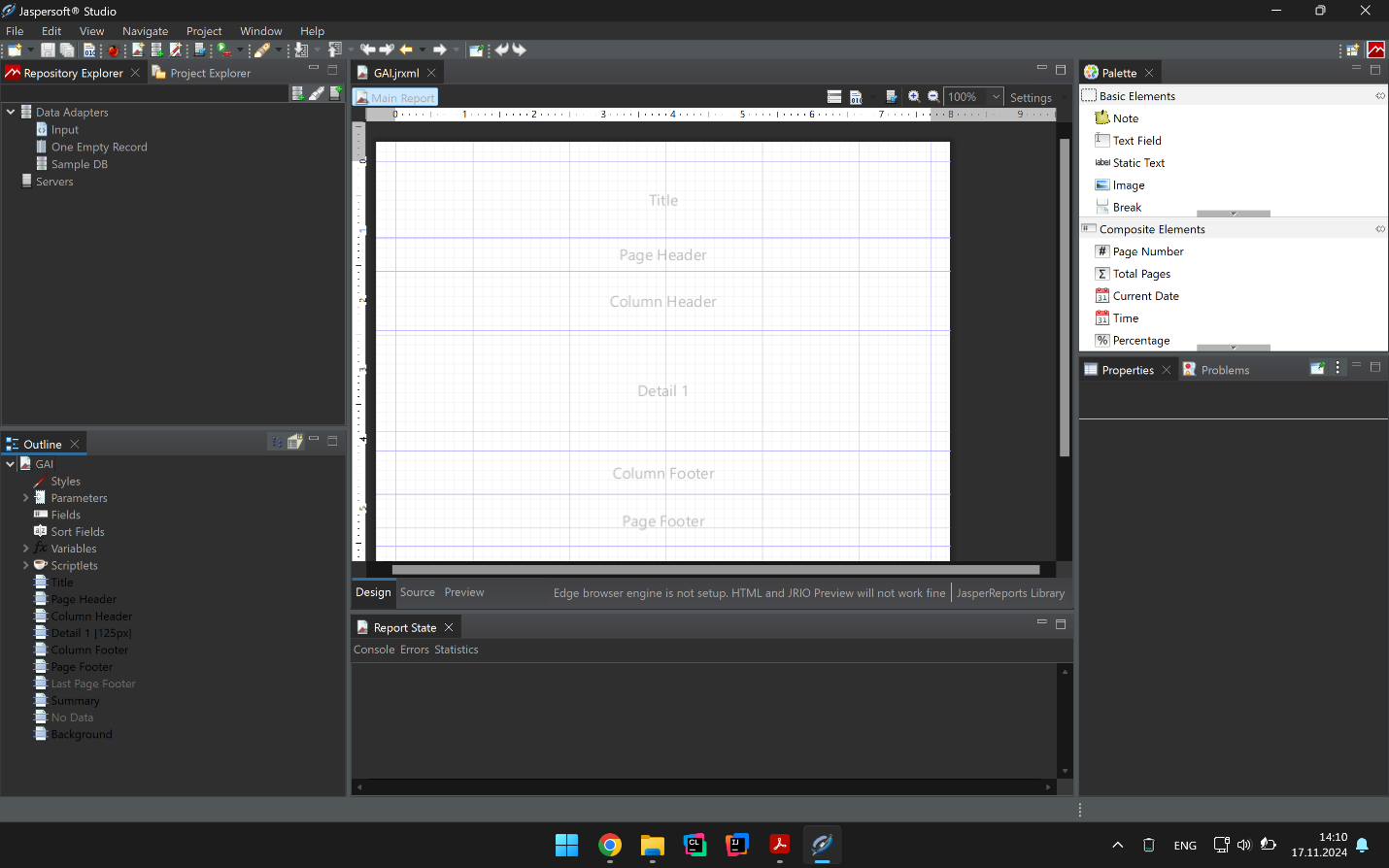


Рисунок 4 – Окно Jaspersoft Studio

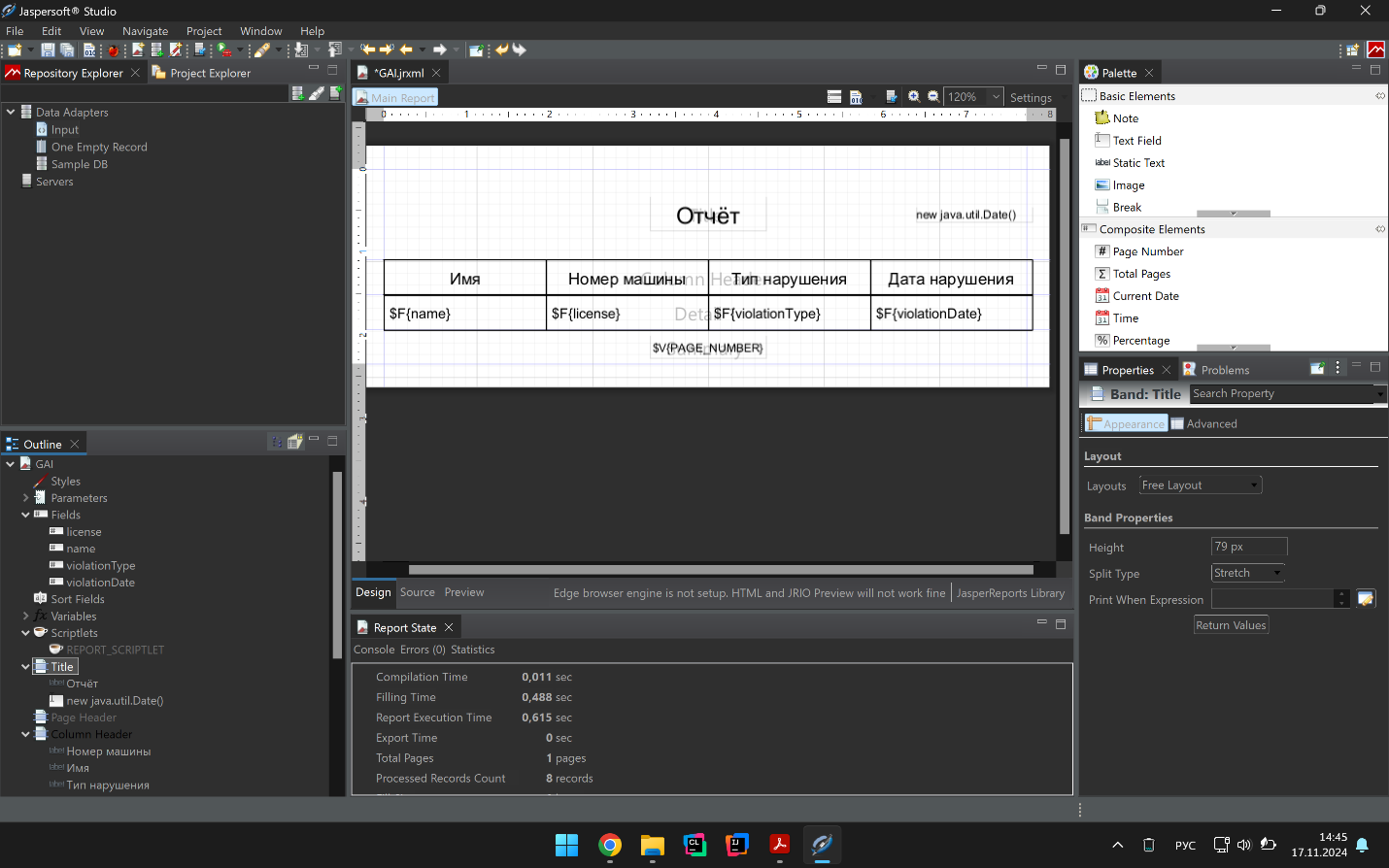


Рисунок 5 – Каркас отчёта

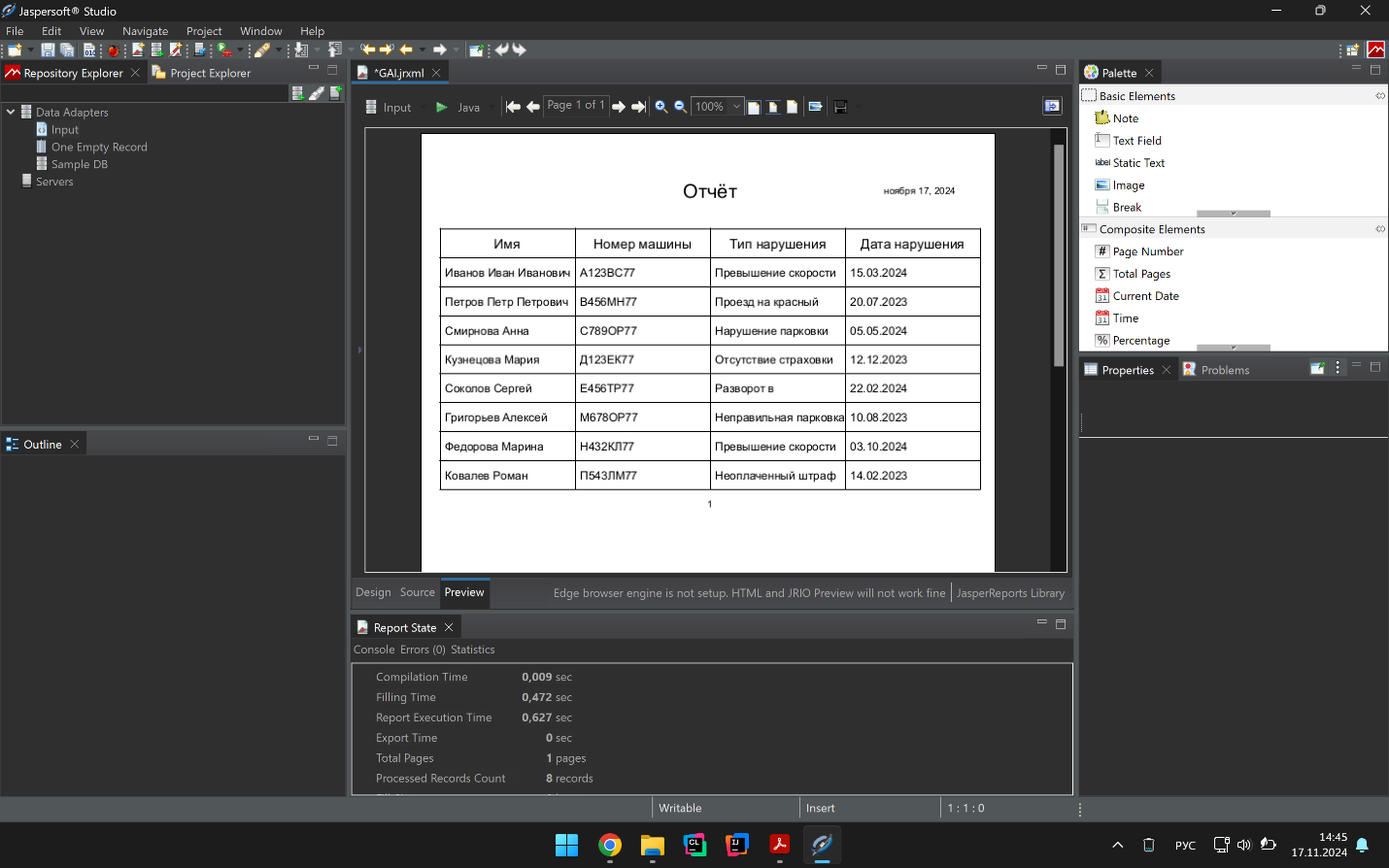


Рисунок 6 – Предосмотр отчёта

# Содержимое файлов



Рисунок 7 – Исходный файл

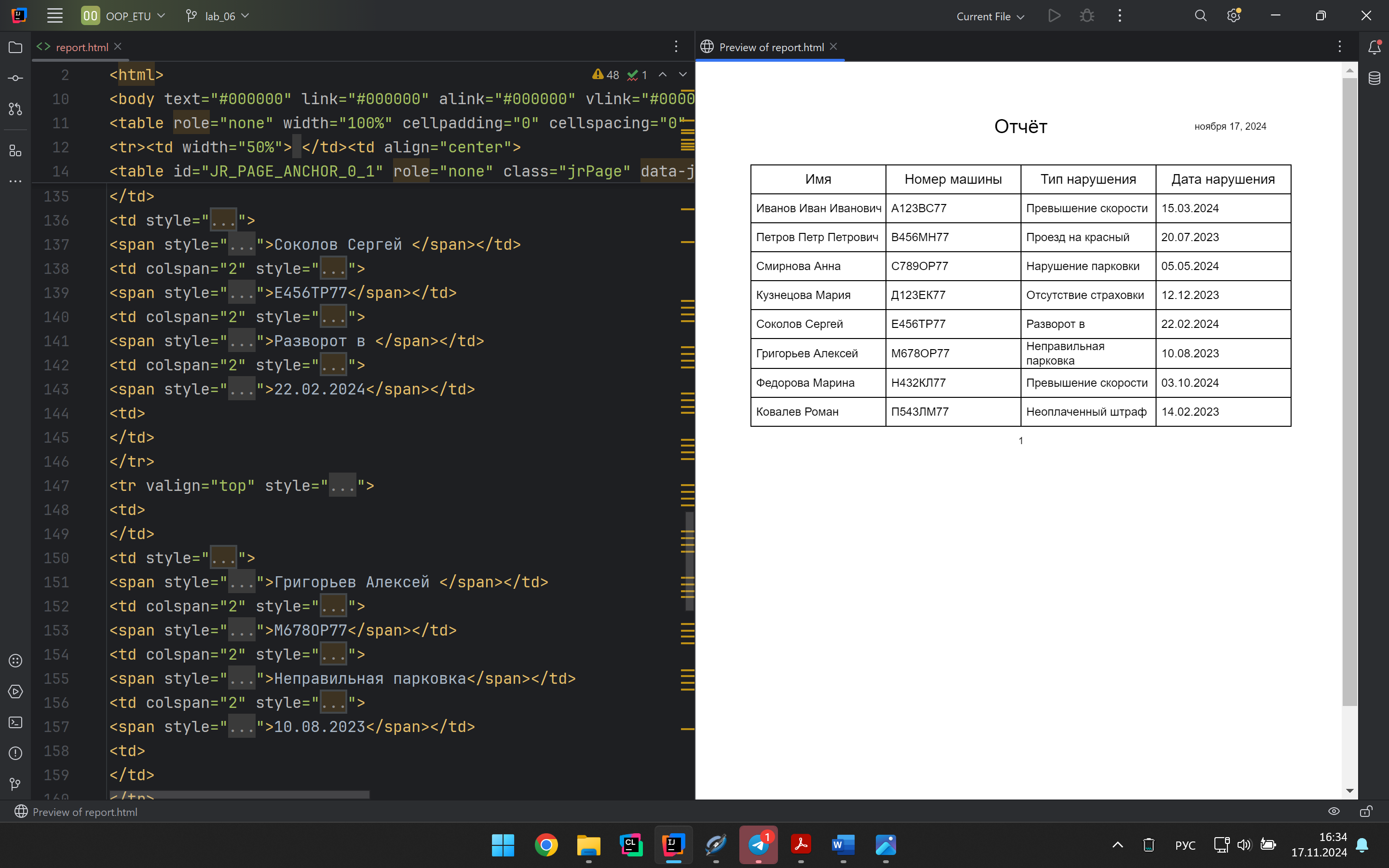


Рисунок 8 – Сгенерированный файл

# Текст программы

Класс Main.java

|  |
| --- |
| **import javax.swing.\*; import javax.swing.table.DefaultTableModel; import javax.swing.filechooser.FileNameExtensionFilter; import java.awt.\*; import java.awt.event.\*; import java.io.\*;  import org.w3c.dom.\*; import javax.xml.parsers.\*; import javax.xml.transform.\*; import javax.xml.transform.dom.\*; import javax.xml.transform.stream.\*;  import org.xml.sax.SAXException;  import Exceptions.\*;  /\*\*  \* Программа для работы с данными о водителях и их нарушениях.  \* Содержит функции добавления, редактирования, удаления записей, а также сохранения и загрузки данных в файл.  \*  \* @author Шарапов Иван 3312  \* @version 1.3  \*/ public class Main {  private JFrame mainFrame;  private DefaultTableModel tableModel;  private JTable dataTable;  private JButton addDriverButton, editDriverButton, deleteDriverButton, loadDriverButton, saveDriverButton;  private JTextField searchField;  private JComboBox<String> searchTypeComboBox;   /\*\*  \* Конструктор класса Main.  \* Инициализирует основное окно приложения для работы с данными.  \*/  public Main() {  // Конструктор по умолчанию, который инициализирует класс Main.  }   /\*\*  \* Метод для создания и отображения главного окна программы.  \* Включает создание таблицы, панели инструментов с кнопками и панель поиска.  \*/  public void show() {  // Создание основного окна приложения  mainFrame = new JFrame("GAI System");  mainFrame.setSize(800, 400);  mainFrame.setLocation(100, 100);  mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   // Создание кнопок для управления записями  addDriverButton = new JButton("Добавить");  editDriverButton = new JButton("Редактировать");  deleteDriverButton = new JButton("Удалить");  loadDriverButton = new JButton("Загрузить");  saveDriverButton = new JButton("Сохранить");   // Панель инструментов, которая содержит кнопки  JToolBar toolBar = new JToolBar("Toolbar");  toolBar.setLayout(new BorderLayout()); // Устанавливаем BorderLayout для панели инструментов   // Панель с кнопками управления (добавление, редактирование, удаление)  JPanel leftPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));  leftPanel.add(addDriverButton);  leftPanel.add(editDriverButton);  leftPanel.add(deleteDriverButton);  toolBar.add(leftPanel, BorderLayout.WEST); // Размещаем в левой части панели   // Панель с кнопками сохранения и загрузки  JPanel rightPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));  rightPanel.add(loadDriverButton);  rightPanel.add(saveDriverButton);  toolBar.add(rightPanel, BorderLayout.EAST); // Размещаем в правой части панели   mainFrame.setLayout(new BorderLayout());  mainFrame.add(toolBar, BorderLayout.NORTH); // Размещение панели инструментов сверху   // Создание таблицы для отображения данных  String[] columns = {"ФИО водителя", "Номер машины", "Дата нарушения", "Тип нарушения"};  String[][] data = {};  tableModel = new DefaultTableModel(data, columns);  dataTable = new JTable(tableModel); // Таблица, которая использует данные tableModel  JScrollPane scrollPane = new JScrollPane(dataTable); // Добавляем прокрутку для таблицы  mainFrame.add(scrollPane, BorderLayout.CENTER); // Размещаем таблицу в центре окна   // Элементы поиска: поле ввода и кнопка "Поиск"  searchTypeComboBox = new JComboBox<>(new String[]{"По имени", "По номеру машины", "По дате нарушения", "По типу нарушения"});  searchField = new JTextField(15);  JButton searchButton = new JButton("Поиск");   JPanel searchPanel = new JPanel();  searchPanel.add(searchTypeComboBox);  searchPanel.add(searchField);  searchPanel.add(searchButton);  mainFrame.add(searchPanel, BorderLayout.SOUTH); // Панель поиска размещается внизу   // Добавляем действия для кнопок   // "Добавить" — действие для добавления новой записи  addDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  new RecordDialog(mainFrame, tableModel, -1).setVisible(true); // -1 означает, что это добавление новой записи  }  });   // "Редактировать" — действие для редактирования выбранной записи  editDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  int[] selectedRows = dataTable.getSelectedRows();   // Проверка, что выбрана ровно одна строка  if (selectedRows.length != 1) {  JOptionPane.showMessageDialog(mainFrame, "Пожалуйста, выберите только одну строку для редактирования.", "Ошибка", JOptionPane.WARNING\_MESSAGE);  return;  }   // Открытие диалогового окна для редактирования выбранной строки  int selectedRow = selectedRows[0];  new RecordDialog(mainFrame, tableModel, selectedRow).setVisible(true);  }  });   // "Удалить" — действие для удаления выбранной записи  deleteDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  // Проверяем, есть ли выделенные строки  int[] selectedRows = dataTable.getSelectedRows();  if (selectedRows.length == 0) {  JOptionPane.showMessageDialog(mainFrame, "Нет выделенных строк для удаления.", "Ошибка", JOptionPane.WARNING\_MESSAGE);  return;  }   // Запрос подтверждения у пользователя  int confirm = JOptionPane.showConfirmDialog(mainFrame, "Вы уверены, что хотите удалить выделенные строки?", "Подтверждение удаления", JOptionPane.YES\_NO\_OPTION);  if (confirm == JOptionPane.YES\_OPTION) {  // Удаляем строки с конца списка, чтобы избежать смещения индексов  for (int i = selectedRows.length - 1; i >= 0; i--) {  tableModel.removeRow(selectedRows[i]);  }  JOptionPane.showMessageDialog(mainFrame, "Выделенные строки успешно удалены.", "Удаление", JOptionPane.INFORMATION\_MESSAGE);  }  }  });   // "Загрузить" — открывает диалоговое окно для выбора файла и загружает данные в таблицу  loadDriverButton.addActionListener(e -> {  JFileChooser fileChooser = new JFileChooser();  fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));  fileChooser.setDialogTitle("Открыть XML файл");  fileChooser.setFileFilter(new FileNameExtensionFilter("XML файлы", "xml"));   int userSelection = fileChooser.showOpenDialog(mainFrame);  if (userSelection == JFileChooser.APPROVE\_OPTION) {  File fileToLoad = fileChooser.getSelectedFile();   try {  // Читаем XML-документ  DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  Document doc = dBuilder.parse(fileToLoad);  doc.getDocumentElement().normalize();   tableModel.setRowCount(0);   NodeList driverNodes = doc.getElementsByTagName("driver");   for (int i = 0; i < driverNodes.getLength(); i++) {  Node node = driverNodes.item(i);  NamedNodeMap attributes = node.getAttributes();   String name = attributes.getNamedItem("name").getNodeValue();  String license = attributes.getNamedItem("license").getNodeValue();  String violationDate = attributes.getNamedItem("violationDate").getNodeValue();  String violationType = attributes.getNamedItem("violationType").getNodeValue();   tableModel.addRow(new Object[]{name, license, violationDate, violationType});  }   JOptionPane.showMessageDialog(mainFrame, "Данные успешно загружены из XML файла.");  } catch (ParserConfigurationException | SAXException | IOException ex) {  JOptionPane.showMessageDialog(mainFrame, "Ошибка при загрузке данных из XML файла.", "Ошибка", JOptionPane.ERROR\_MESSAGE);  ex.printStackTrace();  }  }  });   // "Сохранить" — открывает диалоговое окно для сохранения файла и записывает данные таблицы в файл  saveDriverButton.addActionListener(e -> {  JFileChooser fileChooser = new JFileChooser();  fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));  fileChooser.setDialogTitle("Сохранить как XML");  fileChooser.setFileFilter(new FileNameExtensionFilter("XML файлы", "xml"));   int userSelection = fileChooser.showSaveDialog(mainFrame);  if (userSelection == JFileChooser.APPROVE\_OPTION) {  File fileToSave = fileChooser.getSelectedFile();  if (!fileToSave.getAbsolutePath().endsWith(".xml")) {  fileToSave = new File(fileToSave + ".xml"); // Добавляем расширение .xml, если отсутствует  }   try {  // Создаем XML-документ  DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  Document doc = dBuilder.newDocument();   Element rootElement = doc.createElement("drivers");  doc.appendChild(rootElement);   for (int i = 0; i < tableModel.getRowCount(); i++) {  Element driver = doc.createElement("driver");   driver.setAttribute("name", (String) tableModel.getValueAt(i, 0));  driver.setAttribute("license", (String) tableModel.getValueAt(i, 1));  driver.setAttribute("violationDate", (String) tableModel.getValueAt(i, 2));  driver.setAttribute("violationType", (String) tableModel.getValueAt(i, 3));   rootElement.appendChild(driver);  }   // Сохраняем XML-документ в файл  TransformerFactory transformerFactory = TransformerFactory.newInstance();  Transformer transformer = transformerFactory.newTransformer();  DOMSource source = new DOMSource(doc);  StreamResult result = new StreamResult(fileToSave);  transformer.transform(source, result);   JOptionPane.showMessageDialog(mainFrame, "Данные успешно сохранены в XML файл.");  } catch (ParserConfigurationException | TransformerException ex) {  JOptionPane.showMessageDialog(mainFrame, "Ошибка при сохранении данных в XML файл.", "Ошибка", JOptionPane.ERROR\_MESSAGE);  ex.printStackTrace();  }  }  });   // "Поиск" — выполнает поиск в таблице, по введённой строке  searchButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  try {  validateSearchField(searchField); // Проверка значений в поле поиска  performSearch(searchField.getText()); // Выполнение поиска по введенному тексту  } catch (NullPointerException ex) {  JOptionPane.showMessageDialog(mainFrame, ex.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);  } catch (EmptySearchException ex) {  JOptionPane.showMessageDialog(mainFrame, ex.getMessage(), "Ошибка", JOptionPane.WARNING\_MESSAGE);  }  }  });   mainFrame.setVisible(true); // Делаем главное окно видимым  }   /\*\*  \* Загружает данные из указанного файла в таблицу.  \*  \* @param file Файл, из которого будут загружены данные.  \*/  private void loadDataFromFile(File file) {  try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  String line;  tableModel.setRowCount(0); // Очищаем текущие данные таблицы перед загрузкой новых  while ((line = reader.readLine()) != null) {  // Разделяем строку по табуляциям, чтобы получить значения для каждой колонки  String[] rowData = line.split("\t");  tableModel.addRow(rowData); // Добавляем новую строку в таблицу  }  JOptionPane.showMessageDialog(mainFrame, "Данные успешно загружены!"); // Сообщение об успешной загрузке  } catch (IOException e) {  JOptionPane.showMessageDialog(mainFrame, "Ошибка при загрузке данных."); // Сообщение об ошибке  }  }   /\*\*  \* Сохраняет данные из таблицы в указанный файл.  \*  \* @param file Файл, в который будут сохранены данные.  \*/  private void saveDataToFile(File file) {  try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  // Проходим по каждой строке таблицы  for (int row = 0; row < tableModel.getRowCount(); row++) {  for (int col = 0; col < tableModel.getColumnCount(); col++) {  // Записываем значения ячеек таблицы в файл через табуляцию  writer.write(tableModel.getValueAt(row, col).toString() + "\t");  }  writer.newLine(); // Переход на новую строку после каждой записи  }  JOptionPane.showMessageDialog(mainFrame, "Данные успешно сохранены!"); // Сообщение об успешном сохранении  } catch (IOException e) {  JOptionPane.showMessageDialog(mainFrame, "Ошибка при сохранении данных."); // Сообщение об ошибке  }  }   /\*\*  \* Возвращает индекс столбца для поиска на основе выбранного поля.  \*  \* @param selectedField Поле, выбранное пользователем в JComboBox для поиска  \* @return Индекс столбца для поиска, или -1, если поле не распознано  \*/  private int getColumnIndex(String selectedField) {  switch (selectedField) {  case "По имени":  return 0; // Индекс столбца с ФИО водителя  case "По номеру машины":  return 1; // Индекс столбца с номером машины  case "По дате нарушения":  return 2; // Индекс столбца с датой нарушения  case "По типу нарушения":  return 3; // Индекс столбца с типом нарушения  default:  return -1;  }  }   /\*\*  \* Проверяет значение в поле поиска и генерирует исключения, если поле пустое или содержит null.  \*  \* @param searchField Поле ввода текста для поиска  \* @throws EmptySearchException Если поле ввода пустое  \* @throws NullPointerException Если значение в поле null  \*/  private void validateSearchField(JTextField searchField) throws EmptySearchException, NullPointerException {  String searchText = searchField.getText();  if (searchText == null) {  throw new NullPointerException("Поисковый запрос отсутствует");  }  if (searchText.isEmpty()) {  throw new EmptySearchException();  }  }   /\*\*  \* Выполняет поиск по таблице и выделяет все строки, содержащие указанный текст в выбранном поле, без учета регистра.  \*  \* @param query Строка для поиска  \*/  private void performSearch(String query) {  dataTable.clearSelection(); // Снимаем предыдущее выделение  boolean found = false;   // Приводим запрос к нижнему регистру  String lowerCaseQuery = query.toLowerCase();   // Получаем индекс столбца для поиска  int columnIndex = getColumnIndex((String) searchTypeComboBox.getSelectedItem());  if (columnIndex == -1) {  JOptionPane.showMessageDialog(mainFrame, "Некорректное поле для поиска", "Ошибка", JOptionPane.ERROR\_MESSAGE);  return;  }   // Проходим по всем строкам, но только в выбранном столбце  for (int i = 0; i < tableModel.getRowCount(); i++) {  String cellValue = tableModel.getValueAt(i, columnIndex).toString().toLowerCase();   // Если значение ячейки содержит искомый текст без учета регистра  if (cellValue.contains(lowerCaseQuery)) {  dataTable.addRowSelectionInterval(i, i); // Выделяем строку  found = true;  }  }   if (!found) {  JOptionPane.showMessageDialog(mainFrame, "Совпадения не найдены", "Результат поиска", JOptionPane.INFORMATION\_MESSAGE);  }  }   /\*\*  \* Точка входа в приложение. Запускает метод show() для отображения GUI.  \*  \* @param args Аргументы командной строки (не используются).  \*/  public static void main(String[] args) {  new Main().show(); // Запуск приложения  } }** |

Класс RecordDialog.java

|  |
| --- |
| **import javax.swing.\*; import javax.swing.table.DefaultTableModel; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.time.LocalDate; import java.time.format.DateTimeFormatter; import java.time.format.DateTimeParseException;  /\*\*  \* Диалоговое окно для добавления или редактирования записи в таблице.  \*/ public class RecordDialog extends JDialog {   /\*\*  \* Поле для ввода ФИО водителя.  \*/  private JTextField nameField;  /\*\*  \* Поле для ввода номера машины.  \*/  private JTextField licenseField;  /\*\*  \* Поле для ввода даты нарушения.  \*/  private JTextField dateField;  /\*\*  \* Поле для ввода типа нарушения.  \*/  private JTextField violationField;  /\*\*  \* Модель таблицы, используемая для добавления или обновления записи.  \*/  private DefaultTableModel tableModel;  /\*\*  \* Индекс строки для редактирования, -1 если добавляется новая запись.  \*/  private int rowIndex = -1;   /\*\*  \* Конструктор для создания окна записи (добавление или редактирование).  \*  \* @param parentFrame Родительское окно  \* @param tableModel Модель таблицы  \* @param rowIndex Индекс строки для редактирования (-1 для добавления новой записи)  \*/  public RecordDialog(JFrame parentFrame, DefaultTableModel tableModel, int rowIndex) {  super(parentFrame, rowIndex == -1 ? "Добавить запись" : "Редактировать запись", true);  this.tableModel = tableModel;  this.rowIndex = rowIndex;   setSize(350, 250);  setLocationRelativeTo(parentFrame);  setLayout(new BorderLayout(10, 10));   // Панель с полями ввода  JPanel inputPanel = new JPanel(new GridLayout(4, 2, 5, 5));  nameField = new JTextField(15);  licenseField = new JTextField(10);  dateField = new JTextField(10);  violationField = new JTextField(15);   inputPanel.add(new JLabel("ФИО водителя:"));  inputPanel.add(nameField);  inputPanel.add(new JLabel("Номер машины:"));  inputPanel.add(licenseField);  inputPanel.add(new JLabel("Дата нарушения (ДД.ММ.ГГГГ):"));  inputPanel.add(dateField);  inputPanel.add(new JLabel("Тип нарушения:"));  inputPanel.add(violationField);   // Заполнение полей, если это редактирование  if (rowIndex != -1) {  nameField.setText((String) tableModel.getValueAt(rowIndex, 0));  licenseField.setText((String) tableModel.getValueAt(rowIndex, 1));  dateField.setText((String) tableModel.getValueAt(rowIndex, 2));  violationField.setText((String) tableModel.getValueAt(rowIndex, 3));  }   // Панель с кнопками  JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));  JButton saveButton = new JButton(rowIndex == -1 ? "Сохранить" : "Обновить");  JButton cancelButton = new JButton("Отмена");   saveButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  if (validateAndSave()) {  dispose();  }  }  });   cancelButton.addActionListener(e -> dispose());   buttonPanel.add(saveButton);  buttonPanel.add(cancelButton);   // Добавление панелей в диалоговое окно  add(inputPanel, BorderLayout.CENTER);  add(buttonPanel, BorderLayout.SOUTH);  }   /\*\*  \* Проверяет корректность данных и сохраняет их, если данные верны.  \*  \* @return true, если данные успешно сохранены, иначе false  \*/  private boolean validateAndSave() {  if (nameField.getText().isEmpty() || licenseField.getText().isEmpty() ||  dateField.getText().isEmpty() || violationField.getText().isEmpty()) {  JOptionPane.showMessageDialog(this, "Все поля должны быть заполнены.", "Ошибка", JOptionPane.WARNING\_MESSAGE);  return false;  }   // Проверка формата российского номера  if (!licenseField.getText().matches("^[АВЕКМНОРСТУХ]{1}\\d{3}[АВЕКМНОРСТУХ]{2}\\d{2,3}$")) {  JOptionPane.showMessageDialog(this, "Неверный формат номера. Введите российский номер (например, А123ВС77).", "Ошибка", JOptionPane.WARNING\_MESSAGE);  return false;  }   // Проверка даты нарушения  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");  LocalDate violationDate;  try {  violationDate = LocalDate.parse(dateField.getText(), formatter);  } catch (DateTimeParseException ex) {  JOptionPane.showMessageDialog(this, "Неверный формат даты. Используйте ДД.ММ.ГГГГ.", "Ошибка", JOptionPane.WARNING\_MESSAGE);  return false;  }   LocalDate currentDate = LocalDate.now();  LocalDate fiveYearsAgo = currentDate.minusYears(5);   // Проверка: дата не должна быть в будущем и не старше 5 лет  if (violationDate.isAfter(currentDate)) {  JOptionPane.showMessageDialog(this, "Дата нарушения не может быть в будущем.", "Ошибка", JOptionPane.WARNING\_MESSAGE);  return false;  } else if (violationDate.isBefore(fiveYearsAgo)) {  JOptionPane.showMessageDialog(this, "Нарушение не должно быть старше 5 лет.", "Ошибка", JOptionPane.WARNING\_MESSAGE);  return false;  }   // Сохранение данных  if (rowIndex == -1) {  // Добавление новой записи  tableModel.addRow(new Object[]{nameField.getText(), licenseField.getText(), dateField.getText(), violationField.getText()});  } else {  // Обновление существующей записи  tableModel.setValueAt(nameField.getText(), rowIndex, 0);  tableModel.setValueAt(licenseField.getText(), rowIndex, 1);  tableModel.setValueAt(dateField.getText(), rowIndex, 2);  tableModel.setValueAt(violationField.getText(), rowIndex, 3);  }  return true;  } }** |

Класс EmptySearchException.java

|  |
| --- |
| **package Exceptions;  */\*\*  \* Исключение, вызываемое, когда поле поиска пустое.  \*/* public class EmptySearchException extends Exception {   */\*\*  \* Конструктор без параметров, создающий исключение с сообщением об ошибке пустого поля поиска.  \*/* public EmptySearchException() {  super("Поле поиска не должно быть пустым");  } }** |

# Приложение

Ссылка на репозиторий: <https://github.com/DexTver/OOP_ETU/tree/lab_07>