Курсовой проект "Веб-приложение для визуализации аналитической информации об уровне преступности в регионах России"

Андриянова Анастасия Андреевна 231-361

22 января 2025 г.

1 Введение

Актуальность разработки веб-приложения для визуализации аналитической информации об уровне преступности в регионах России обусловлена необходимостью более глубокого понимания ситуации в сфере безопасности и преступности на территории страны. На фоне растущих угроз и вызовов, связанных с общественным порядком, важно иметь доступ к систематизированной информации, которая позволяет анализировать статистику зарегистрированных и расследованных преступлений по регионам.

Проблемой является недостаток доступных инструментов для комплексного анализа таких данных. Существующие подходы, например, исследования на основе отчетов Росстандарта, имеют свои ограничения, такие как невозможность динамичного отображения данных по регионам и сравнения соотношения загеристрированных и раскрытых преступлений в удобной визуальной форме.

В связи с этим, разрабатываемое веб-приложение представляет собой удобный инструмент для визуализации и анализа статистических данных о преступности в России. Приложение позволяет эффективно сравнивать уровень преступности по регионам и отслеживать динамику изменений.

2 Цель и задачи работы

Цель работы заключается в создании веб-приложения, которое будет предоставлять пользователю доступ к аналитической информации о пре-

ступности в различных регионах России, с возможностью интерактивной визуализации данных. Для достижения этой цели, поставлены следующие задачи:

- Разработка интерфейса приложения, обеспечивающего удобное взаимодействие с данными.
- Обработка и подготовка исходных данных для отображения на графиках.
- Реализация функционала сортировки по регионам.
- Внедрение алгоритмов для обработки данных и их вывода в виде визуальных графиков.

Исходные данные для приложения в форме открытых данных — это статистические данные по зарегистрированным и расследованным преступлениям, предоставленные официальным источником Министерства внутренних дел Российской Федерации. Датасет состоит из таблицы, описывающей сводку данных по категориям преступлений в разных регионах РФ и их количество.

Для разработки использованы технологии Java(Основной язык программирования, используемый для разработки всего приложения. Предоставляет объектно-ориентированный подход, надежность и платформенную независимость.), Spring Boot (Фреймворк для создания веб-приложений. Используется для: Упрощенной работы с зависимостями. Организации архитектуры приложения с минимальными конфигурациями.), базы данных MySQL (Система управления базами данных, используется для хранения данных о преступлениях, регионах и их статусах, обеспечения быстрого доступа и манипуляции с данными через SQL-запросы.) и библиотека для визуализации данных JFreeChart (Инструмент для создания графиков и диаграмм. Используется для генерации круговых диаграмм (Pie Chart) для визуализации статистики преступлений, преобразования графиков в изображения, которые затем отображаются в веб-интерфейсе.).

3 Проектирование приложения

3.1 Функциональные возможности:

Веб-приложение должно включать следующие функциональные возможности:

- Фильтрация данных по регионам.
- Визуализация статистики преступлений в виде графиков.
- Возможность наглядного сравнения зарегистрированных и расследованных преступлений между собой для каждого региона.

3.2 Основные модули:

Для этого приложение будет включать следующие основные модули:

- Модуль обработки данных — для загрузки и предобработки исходных данных.
- Модуль визуализации — для наглядного отображения графиков и диаграмм с целью упрощения восприятия информации.
- Модуль пользовательского интерфейса — для взаимодействия с пользователем, для улучшения общей организации пользовательского интерфейса, а также для упрощения процесса разработки.
- База данных — для хранения статистики преступлений по регионам, категориям "зарегистрированные"и "расследованные статьям УК РФ и количеству происшествий за год.

3.3 Модель обрабатываемых данных

Структура таблицы Statistics, с которой производилось взаимодействие в ходе проекта, представлена на Рисунке 1.

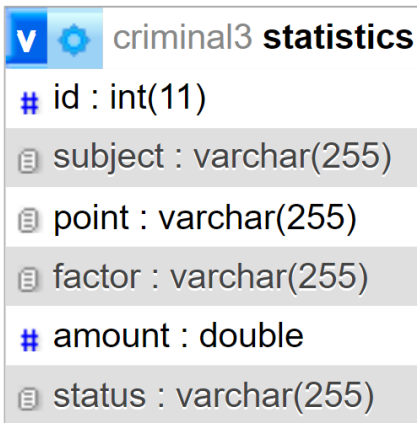

Рис. 1: Модель таблицы Statistics

- **ID**: Уникальный идентификатор.
- **Subject**: Субъект (регион) Российской Федерации.
- **Point**: Пункт ФПСР.
- **Factor**: Наименование статистического показателя.
- **Amount**: Значение статистического показателя (за январь - декабрь 2024 г.).
- **Status**: Статус стадии рассмотрения преступления.

Вырезка из таблицы Statistics представлена на Рисунке 2.

id	subject	point	factor	amount	status
1	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	1669	зарегистрирован
2	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	3222	зарегистрирован
3	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	654	зарегистрирован
4	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	166418	зарегистрирован
5	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	7118	зарегистрирован
6	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	1053	зарегистрирован
7	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	1631	зарегистрирован
8	Центральный фед.округ	3_3	Количество преступлений, зарегистрированных в отче...	459	зарегистрирован
9	Центральный фед.округ	3_3	Количество предварительно расследованных преступле...	1248	расследован
10	Центральный фед.округ	3_3	Количество предварительно расследованных преступле...	2993	расследован
11	Центральный фед.округ	3_3	Количество предварительно расследованных преступле...	608	расследован
12	Центральный фед.округ	3_3	Количество предварительно расследованных преступле...	61064	расследован

Рис. 2: Таблица Statistics

3.4 Схема сайта

Схема сайта представлена на Рисунке 3, представляет собой обобщенную визуализацию элементов на главной странице.

3.5 Диаграмма прецедентов

Диаграмма прецедентов демонстрирует функциональные возможности взаимодействия пользователя и администратора приложения. Диаграмма представлена на Рисунке 4.

Например, пользователь может выбирать любой субъект Российской Федерации и ознакомиться со статистикой преступлений по интересующему региону.

Рис. 3: Схема сайта

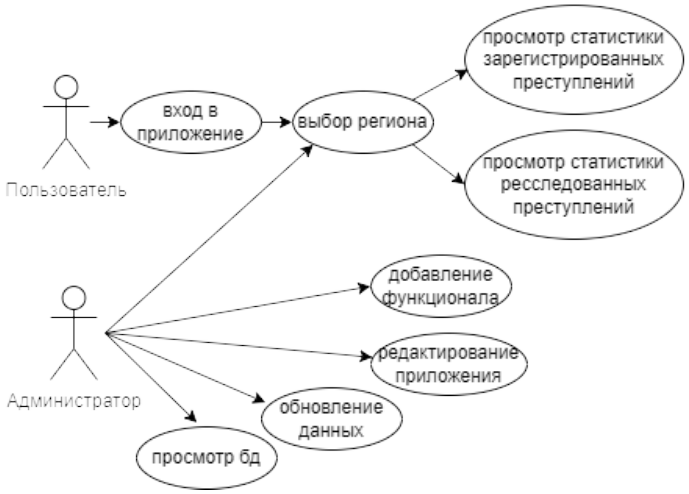

Рис. 4: Диаграмма прецедентов

3.6 Диаграмма классов

Диаграмма классов предназначена для представления модели статической структуры программной системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов представлена на Рисунке 5, см. в разделе Приложение.

3.7 Процедура предобработки данных.

Процедура предобработки данных в коде включает:

1. Группировку преступлений по регионам.

2. Сортировку регионов.

3. Разделение преступлений на группы по статусу (зарегистрированные/расследованные).

4. Подготовку данных для визуализации (графики).

Далее каждый пункт предобработки данных будет рассмотрен более подробно.

3.7.1 Группировка преступлений по регионам

В методе `showCriminals` контроллера данные преступлений группируются по полю `subject` (регион):

- Используется `Map<String, List<Criminal» groupedCriminals` для хранения преступлений, сгруппированных по регионам.
- Для каждого региона создается запись, если её нет в `Map`.

Процедура:

1. Для каждого объекта `Criminal` проверяется, существует ли соответствующий ключ в `groupedCriminals`.
2. Если ключ отсутствует, создается новый список преступлений для данного региона.
3. Преступление добавляется в соответствующий список.

   Группировка данных преступлений по регионам представлена в Листинге 1.

```
1  public String showCriminals(Model model) throws IOException {
2      List<Criminal> criminalList = criminalService.getAll();
3      Map<String, List<Criminal>> groupedCriminals = new
           HashMap<>();
4      for(Criminal criminal : criminalList) {
5          String subject = criminal.subject;
6          if (!groupedCriminals.containsKey(subject)) {
7              groupedCriminals.put(subject, new ArrayList<>());
8          }
9          groupedCriminals.get(subject).add(criminal);
10     }
11     String[] regions = criminalService.getAllSubjects();
12     String[] sortedRegions = Arrays.stream(regions)
```

```
13              .sorted(Comparator.comparing(s -> s.substring(0,
                    1).toLowerCase(Locale.ROOT)))
14              .toArray(String[]::new);
15      regions = sortedRegions; //

16      List<CriminalList> regionWithCrimesRegAndDone = new
            ArrayList<>();
17      for (int i = 0; i < regions.length; i++) {
18          CriminalList c = new CriminalList(regions[i],
                groupedCriminals.get(regions[i]));
19          c.setGraphReg(showGraph(c.getCriminalsReg(), c.
                getSubject()));
20          c.setGraphDone(showGraph(c.getCriminalsDone(), c.
                getSubject()));
21          regionWithCrimesRegAndDone.add(c);
22      }
23      model.addAttribute("regionWithCrimesRegAndDone",
            regionWithCrimesRegAndDone);
24      model.addAttribute("regions", regions);
25      return "criminal";
26  }
```

Листинг 1: Метод showCriminals

3.7.2 Сортировка регионов

Список регионов сортируется по алфавиту для удобства отображения пользователю:

```
1  String[] sortedRegions = Arrays.stream(regions)
2      .sorted(Comparator.comparing(s -> s.substring(0, 1).
           toLowerCase(Locale.ROOT)))
3      .toArray(String[]::new);
```

Листинг 2: Сортировка по регионам

Процедура:

1. Используется `stream()` для преобразования массива регионов в поток данных.

2. Региональные названия приводятся к нижнему регистру для корректной сортировки.

3. Поток сортируется с помощью компаратора и преобразуется обратно в массив.

3.7.3 Группировка преступлений по статусу

Методы `processReg` и `processDone` в классе `CriminalProcessor`:

- Используются для разделения списка преступлений на две группы: «зарегистрированные» и «расследованные».
- Реализованы с использованием `Stream API` для группировки данных по полю `status`.

Методы для обработки списка преступлений по статусу представлены в Листинге 3.

```
1  public static List<Criminal> processReg(List<Criminal>
       criminals) {
2      Map<String, List<Criminal>> groupedSalaries = criminals.
           stream()
3              .collect(Collectors.groupingBy(Criminal::
                   getStatus));
4      List<Criminal> regcriminals = groupedSalaries.get("
                                         ");
5      return regcriminals;
6  }
7
8  public static List<Criminal> processDone(List<Criminal>
       criminals) {
9      Map<String, List<Criminal>> groupedSalaries = criminals.
           stream()
10             .collect(Collectors.groupingBy(Criminal::
                   getStatus));
11
12     List<Criminal> regcriminals = groupedSalaries.get("
                                  ");
13     return regcriminals;
14 }
```

Листинг 3: Методы для обработки списка преступлений

Процедура:

1. Исходный список преступлений (`List<Criminal>`) обрабатывается с помощью метода `collect(Collectors.groupingBy())`, где ключом является `status` преступления.
2. Из группированных данных выбираются две категории:
   - «зарегистрирован» для зарегистрированных преступлений.
   - «расследован» для расследованных преступлений.

3. Возвращаются два отдельных списка преступлений для дальнейшего анализа.

3.7.4 Подготовка данных для графиков

Метод `showGraph` формирует данные для визуализации в виде круговой диаграммы (Pie Chart):

- **Метки (labels):** Основные категории преступлений (например, «Убийство», «Кража»).
- **Значения (values):** Количество преступлений для каждой категории.

Метод для генерации графика представлен в Листинге 4.

```
1  public String showGraph(List<Criminal> criminals, String name
       ) throws IOException {
2      List<String> labels = Arrays.asList("");
3      List<Double> values = new ArrayList<>();
4      for (int i = 0; i < criminals.size(); i++) {
5          values.add(criminals.get(i).getAmount());
6      }
7
8      ByteArrayOutputStream baos = new ByteArrayOutputStream();
9
10     ChartUtils.writeChartAsPNG(baos,
11                                SimpleGraphGenerator.
                                       generatePieChart(labels,
                                       values, name),
12                                600, 400);
13
14
15     byte[] imageBytes = baos.toByteArray();
16
17     String base64Image = Base64.getEncoder().encodeToString(
           imageBytes);
18     String str = "" + base64Image;
19     return str;
20 }
```

Листинг 4: Метод для генерации графика

Процедура:

1. Создается список меток категорий.
2. Для каждой категории из списка преступлений извлекаются значения (`amount`) и добавляются в `values`.

3. На основе данных генерируется диаграмма, преобразованная в формат `PNG`.

4. Изображение кодируется в формат `Base64` для отображения в веб-интерфейсе.

4 Реализация приложения

Реализация проекта включает создание серверной части на Spring Boot, выгрузку и обработку базы данных для хранения статистики, а также интеграцию с библиотеками для визуализации данных. Описание классов представлено ниже.

4.1 Criminal

Это класс модели, который представляет собой строку в таблице statistics. Он используется для сохранения информации о преступлениях. Листинг кода для класса Criminal представлен в Листинге 5.

```
1  @Entity
2  @Data
3  @Table(name = "statistics")
4  @AllArgsConstructor
5  @NoArgsConstructor
6  public class Criminal {
7      @Id
8      @Column(name="id")
9      private int id;
10
11     @Column(name = "subject")
12     public String subject;
13
14     @Column(name = "point")
15     private String point;
16
17     @Column(name = "factor")
18     private String factor;
19
20     @Column(name = "amount")
21     private Double amount;
22
23     @Column(name = "status")
24     private String status;
25 }
```

Листинг 5: Класс Criminal

Этот класс использует аннотации JPA (`@Entity` (Entity в Java — это класс, который представляет данные из таблицы базы данных, упрощают взаимодействие с базой данных, предоставляя способ работы с данными через объекты, а не через SQL-запросы.), `@Column` и т.д.) для связи с базой данных. Он является объектом, который будет сохранен и извлечен из базы данных.

4.2 CriminalRepository

Это интерфейс, который расширяет `JpaRepository`. Он предоставляет методы для работы с базой данных: сохранение, поиск, обновление объектов `Criminal`. Методы интерфейса CriminalRepository представлены в Листинге 6.

```
1 @Repository
2 public interface CriminalRepository extends JpaRepository<
      Criminal, Integer> {
3     List<Criminal> findById(int id);
4     List<Criminal> findBySubject(String subject);
5 }
```

Листинг 6: Интерфейс CriminalRepository

Методы:

- `findById(int id)` и `findBySubject(String subject)` используются для извлечения преступлений по их идентификатору или категории (`subject`).

Репозитории автоматически реализуют стандартные CRUD операции для сущностей.

4.3 CriminalService

Это сервисный класс, который использует `CriminalRepository` для извлечения данных из базы данных. Он инкапсулирует логику работы с данными и обеспечивает их обработку для контроллеров. Наполнение класса представлено в Листинге 7.

```
1 private final CriminalRepository criminalRepository;
2 public CriminalService(CriminalRepository criminalRepository)
      {
3     this.criminalRepository = criminalRepository;
4 }
5 public List<Criminal> getAll() {
6     return criminalRepository.findAll();
```

```
7  }
8  public String[] getAllSubjects() {
9      List<Criminal> cr = criminalRepository.findAll();
10     List<String> uniqueRegion = new ArrayList<>();
11     for (int i = 0; i < cr.size(); i++) {
12         uniqueRegion.add(cr.get(i).getSubject());
13     }
14     return new HashSet<>(uniqueRegion).toArray(String[]::new)
           ;
15 }
```

Листинг 7: Класс CriminalService

Методы:

- `getAll()`: получает все преступления из базы данных.
- `getAllSubjects()`: получает все категории преступлений.

Этот класс также обрабатывает данные перед их отправкой в контроллер.

4.4 Controller

Контроллер обрабатывает запросы от пользователя и управляет отображением данных в модели. Метод showCriminals() продемонстрирован в Листинге 1.

Метод showCriminals():

- Извлекает все преступления с помощью `criminalService.getAll()`.
- Группирует преступления по статусу с помощью `groupedCriminals.put(subject, new ArrayList<>())`.
- Получает две категории статуса преступлений с помощью `criminalService.getAllSub`
- Для каждой категории генерирует два графика (для зарегистрированных преступлений и завершенных дел) с помощью метода `showGraph()`.
- Затем добавляет данные в модель, которая передается в представление (например, для отображения на веб-странице).

Метод showGraph():

- Создает график в формате PNG с использованием библиотеки `JFreeChart`. Он использует список преступлений и их количества (`amount`) для создания круговой диаграммы.

- После генерации графика он преобразует его в строку формата base64, чтобы передать изображение в HTML как встроенную картинку.

Метод showGraph() представлен в Листинге 4 Подготовка данных для графиков.

4.5 CriminalList

Этот класс используется для организации данных о преступлениях по категориям. Он помогает разделить преступления на две группы: зарегистрированные (`criminalsReg`) и завершенные (`criminalsDone`). Атрибуты класса CriminalList представлены в Листинге 8.

Атрибуты:

```
1 private String graphReg;
2 private String graphDone;
3 private String subject;
4 private List<Criminal> criminalsReg;
5 private List<Criminal> criminalsDone;
6 private int size = 0;
```

Листинг 8: Атрибуты класса CriminalList

Методы:

- `getCriminalsReg()` и `getCriminalsDone()` возвращают соответствующие списки преступлений.
- `setGraphReg()` и `setGraphDone()` устанавливают графики для каждой категории.
- `getGraphReg()` и `getGraphDone()` возвращают строки, содержащие base64-encoded изображения графиков.

4.6 CriminalProcessor

Этот класс занимается обработкой данных о преступлениях. Он разделяет преступления на зарегистрированные и завершенные (методы `processReg` и `processDone`), чтобы отделить их для дальнейшего анализа и отображения. Листинг 3 Группировка преступлений по статусу отображает наполнение класса.

4.7 Criminal3Application

Этот класс запускает Spring Boot приложение. Аннотация `@SpringBootApplication` активирует автоматическую настройку Spring и сканирование компонентов. Программный код представлен в Листинге 9.

```
1 @SpringBootApplication
2 public class Criminal3Application {
3     public static void main(String[] args) {
4         SpringApplication.run(Criminal3Application.class,
              args);
5     }
6 }
```

Листинг 9: Главный класс Criminal3Application

4.8 SimpleGraphGenerator

Этот класс отвечает за создание круговых диаграмм с использованием библиотеки JFreeChart.

Метод generatePieChart оздает набор данных (`PieDataset`) из списков меток и значений. Генерирует круговую диаграмму с заданным названием. Метод generatePieChart продемонстрирован в Листинге 10.

```
1  public class SimpleGraphGenerator {
2      public static JFreeChart generatePieChart(List<String>
           labels, List<Double> values, String name) throws
           IOException {
3          PieDataset dataset = createDataset(labels, values);
4          JFreeChart chart = ChartFactory.createPieChart(
5                  name,
6                  dataset,
7                  true,
8                  true,
9                  false
10         );
11         customizeChart(chart);
12         return chart;
13     }
14 }
```

Листинг 10: SimpleGraphGenerator - Метод generatePieChart

Метод customizeChart Настраивает внешний вид диаграммы: шрифт, цвета, контуры, отображение процентов. Демонстрация работы представлена в Листинге 11.

```
1 private static void customizeChart(JFreeChart chart) {
```

```
2       PiePlot plot = (PiePlot) chart.getPlot();
3       plot.setBackgroundPaint(Color.WHITE);
4       plot.setLabelFont(new Font("Arial", Font.PLAIN, 12));
5       plot.setLabelPaint(Color.DARK_GRAY);
6       PieSectionLabelGenerator labelGenerator = new
            StandardPieSectionLabelGenerator(
7               "{0} ({1})",
8               NumberFormat.getNumberInstance(),
9               NumberFormat.getNumberInstance()
10      );
11      plot.setLabelGenerator(labelGenerator);
12      plot.setSectionOutlinesVisible(true);
13      plot.setOutlinePaint(Color.LIGHT_GRAY);
14      plot.setOutlineStroke(new BasicStroke(1.0f));
15
16      chart.getTitle().setFont(new Font("Arial Black", Font.
            BOLD, 18));
17      chart.getTitle().setPaint(Color.DARK_GRAY);
18  }
```

Листинг 11: Метод настройки диаграммы (customizeChart)

4.9 Maven (pom.xml)

Это конфигурация для Maven, которая используется для сборки и управления зависимостями вашего Spring Boot проекта.

Основные части:

Spring Boot:

- В разделе `<parent>` указан родительский проект `spring-boot-starter-parent`, который предоставляет основные настройки для Spring Boot.
- В разделе `<dependencies>` добавлены необходимые зависимости для работы с базой данных (JPA), веб-сервисами (Spring Web), а также с FreeMarker для шаблонов.

Зависимости:

- `spring-boot-starter-data-jpa`: Для работы с JPA и взаимодействия с базой данных.
- `spring-boot-starter-freemarker`: Для использования FreeMarker в качестве шаблонного движка (создание HTML-шаблонов на сервере).
- `spring-boot-starter-web`: Для работы с веб-приложением (REST API, контроллеры).

- `mysql-connector-j`: Драйвер для подключения к базе данных MySQL.
- `spring-boot-starter-test`: Для тестирования.
- `jakarta.persistence-api`: Для работы с JPA.
- `lombok`: Для упрощения работы с Java (например, генерация геттеров и сеттеров).
- `jfreechart`: Для создания диаграмм (вероятно, используется на серверной стороне для генерации графиков).

5 Основные сценарии использования приложения

Приложение будет включать следующие пользовательские сценарии:

- Выбор региона: Пользователь может выбрать интересующий его регион из списка и увидеть графики с данными о преступлениях.
- Сравнение данных по зарегистрированным и расследованным преступлениям: Пользователь может выбрать регион для сравнения статистики на графиках.
- Пользователь может открыть мобильную версию приложения без искажения функционала и с корректным отображением графиков.

Пример использования представлен на Рисунке 6 и 7. См. в разделе Приложение.

6 Заключение

Заключение

Результаты разработки веб-приложения для визуализации аналитической информации о преступности в регионах России подчеркивают важность создания современных инструментов для анализа и интерпретации сложных данных. Разработанное приложение предоставляет пользователям удобный интерфейс для сравнения данных о преступности по различным регионам, а также наглядные графические представления информации. Это делает его полезным инструментом для органов государственной власти, аналитиков, исследователей и всех, кто заинтересован в повышении уровня общественной безопасности.

Приложение способствует выявлению ключевых тенденций и проблемных зон, что, в свою очередь, может способствовать улучшению качества принимаемых решений и реализации эффективных мер профилактики преступности. Визуализация данных с использованием круговых диаграмм помогает упростить восприятие сложных статистических данных и делает их доступными для более широкой аудитории.

Несмотря на достигнутые результаты, проект имеет значительный потенциал для дальнейшего развития. Возможные направления расширения включают добавление новых аналитических функций, таких как прогнозирование тенденций преступности с использованием методов машинного обучения, интеграция дополнительных источников данных для повышения точности анализа, а также улучшение визуализации за счет применения более интерактивных графиков и карт.

Таким образом, данный проект представляет собой не только полезный инструмент для анализа данных, но и платформу для дальнейших исследований и разработок в области визуализации и анализа данных о преступности. Его реализация способствует повышению прозрачности информации и созданию условий для более эффективного управления безопасностью в стране.

К данной пояснительной записке прилагается на ссылка на репозиторий.

7 Список литературы и интернет-ресурсов

1. ГОСТ Р 7.0.5-2008. Система стандартов по информации, библиотечному и издательскому делу. https://diss.rsl.ru/datadocs

2. Министерство внутренних дел Российской Федерации. Открытые данные : стат. отчеты об уровне преступности [Электронный ресурс]. – URL: https://мвд.рф/открытые-данные

3. Spring Boot. Официальная документация [Электронный ресурс]. – URL: https://spring.io/projects/spring-boot

4. Уголовный кодекс Российской Федерации : [федер. закон от 13 июня 1996 г. № 63-ФЗ]. – Москва https://diss.rsl.ru/datadocs

5. Министерство торговли США. Java SE 8 Documentation. The Java™ Tutorials [Электронный ресурс]. – URL: https://docs.oracle.com/javase/tutorial

6. JPA — Java Persistence API [Электронный ресурс]. – URL: https://javarush.com/groups/p
jpa

8 Приложение

В приложении приведены диаграммы и схемы, на которые есть отсылка из основного текста.

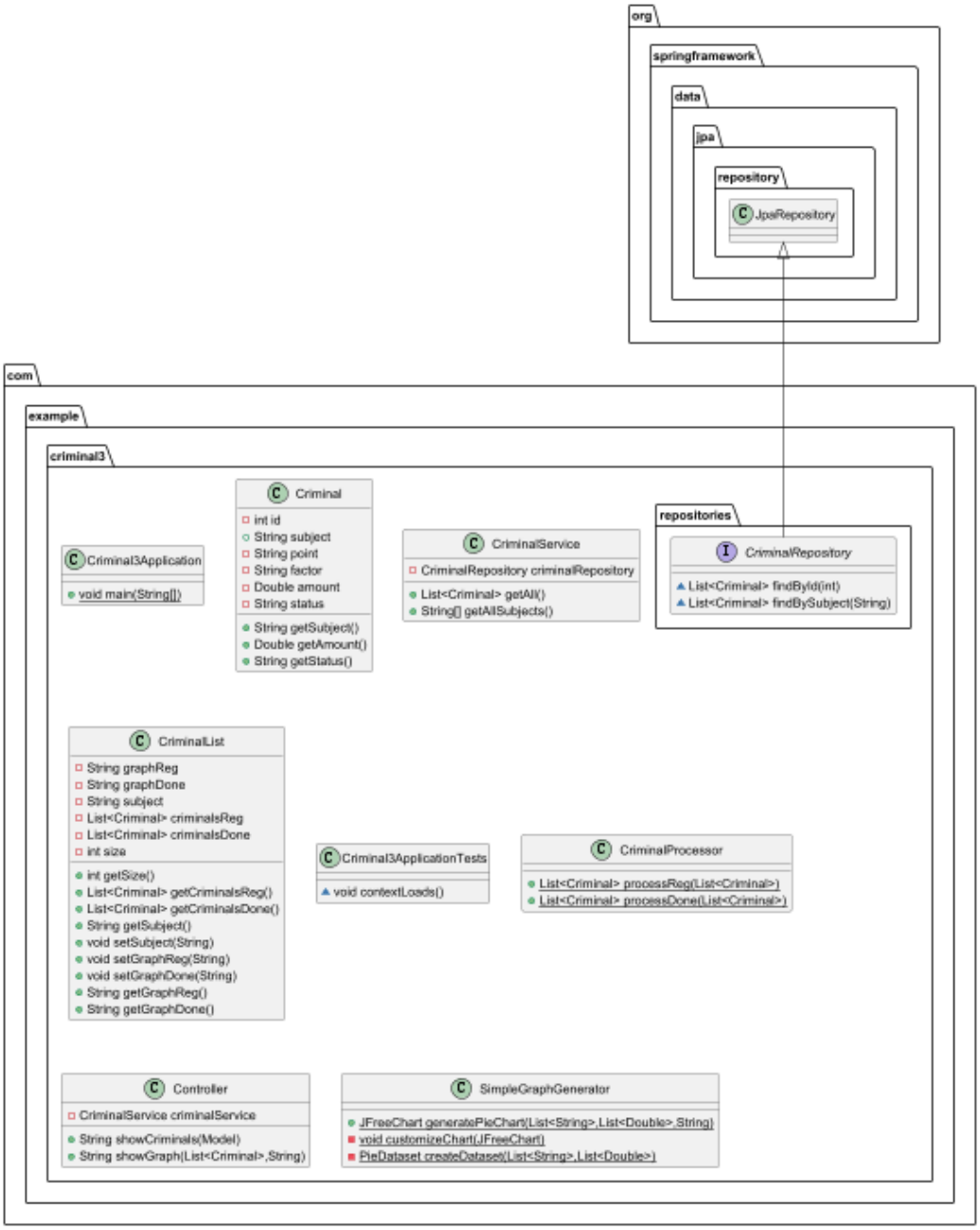

Рис. 5: Диаграмма классов

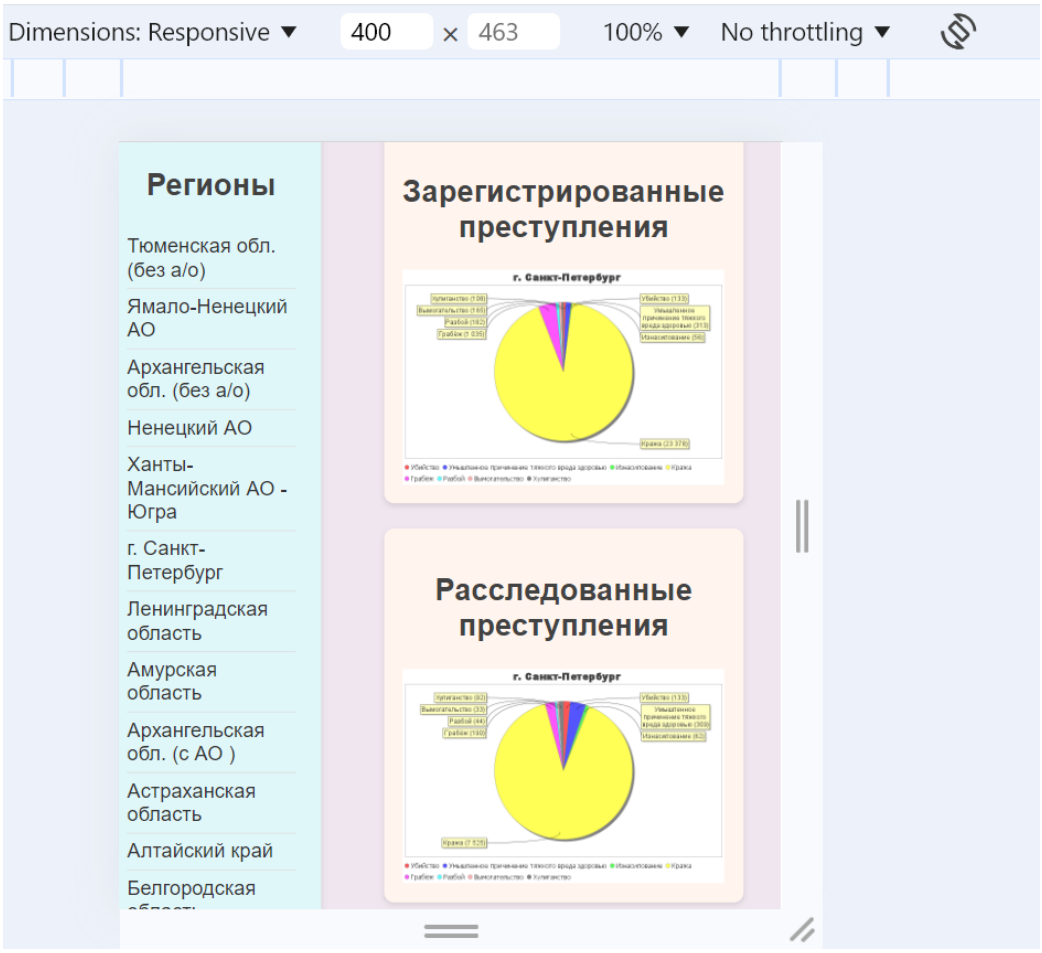

Рис. 6: Пример использования мобильной версии приложения

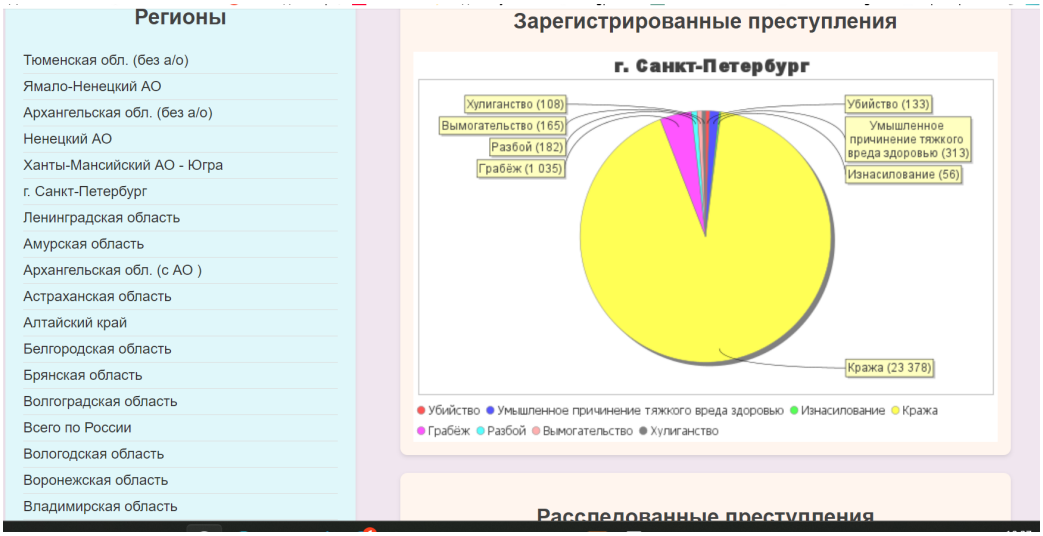

Рис. 7: Пример использования веб-версии приложения