

Ministerul Educației și Cercetării al Republicii Moldova
IP Centrul de Excelență în Informatică și Tehnologii Informaționale

RAPORTUL STAGIULUI DE PRACTICĂ

Elevul: **Rusanov Nichita**

Grupa: **P-1944R**

Specialitatea: **Programare și analiza produselor program**

Baza de practică: **Cedacri International SRL, mun. Chișinău,
Republica Moldova**

Conducătorul stagiului de practică
de la unitatea economică

Malic Lucia

Conducătorul stagiului de practică
de la Centrul de Excelență

Pasecinic Irina

Chișinău, 2023

Введение	4
Описание практики	4
Просмотр приложений:	5
Содержание действий и задания.....	5
1. Data Structures Calculator.....	5
• Продумывание работы алгоритма.....	5
• Сам алгоритм:	6
2. Maven	8
• Разбиение калькулятора на классы и модули:	8
• Добавление зависимостей для модулей и самих модулей.....	9
○ parent pom.xml:	9
○ app pom.xml:.....	9
3. SQL.....	9
• Прохождение задач и тестов на сайте sqlzoo.net:	9
4. Jasper Reports.....	10
• Созданный отчет в графическом редакторе Jasper Studio:.....	10
○ JRMapCollectionDataSource:	12
○ JRBeanCollectionDataSource:	12
○ JRResultSetCollectionDataSource:	13
○ Dynamic Report.....	14
○ Crosstab Report	15
5. Junit.....	16
• Создание домена.....	16
• Junit тесты для моего приложения:	16
• Конвертер для того, чтобы заполнить базу данных используя файл CSV.....	20
6. Clean Code + Refactoring.....	22
7. Hibernate	22
• Конфигурация Hibernate для подключения к базе данных:	23
• Создания моделей, соответствующих таблицам из БД:.....	23
○ Author (Модель):.....	23
○ Book (Модель):.....	23
• Создание репозитория (DAO) для доступа к CRUD операциям с БД:.....	24
○ AuthorDAO:	24
○ BookDAO:	24
• Создание тестов для проверки работоспособности всех методов из сервисов:.....	24
○ Author Service тесты:.....	24

○ Book Service тесты:.....	27
8. Spring	29
• Конфигурация Spring	29
• Создание моделей (Entity) для их связи с БД:.....	29
○ Author:	29
○ Book:.....	30
• Создание DAO и наследование JpaRepository от Spring	30
• Добавление сервисов	30
○ AuthorService:.....	31
○ BookService:.....	32
• Написание тестов с помощью технологии Mockito	32
○ AuthorService тест:	32
○ BookService тест:	35
9. Создание веб приложения используя Spring	37
• Выбор технологий для написания приложения	37
• Конфигурация Spring проекта:.....	37
• Определение и создание Entity базы данных в Spring.....	37
○ Message Entity	37
○ Room Entity	38
○ UserEntity.....	38
• Конфигурация WebSoket для передачи сообщений с его помощью	39
• Конфигурация SpringSecurity для реализации авторизации и регистрации	39
• Создание DAO для работы с базой данных в Spring.....	41
○ MessageDAO	41
○ RoomDAO	41
○ UserDAO.....	42
• Распределение всей логики на эндпоинты (Mappings)	42
○ Authentication controller для регистрации и логина	42
○ Message Controller для работы с сообщениями пользователей.....	43
○ Rooms Controller для работы с комнатами при помощи нужных запросов	45
○ Users Controller для получения информации о пользователях	45
• Создание UI логин формы	46
• Создание формы регистрации по принципу логин формы	46
• Создание UI и логики самого чата.....	47
○ chat.html	47
○ custom.js	53

○ chat.js	57
• Тестирование и демонстрация приложения:.....	60
○ Регистрация	60
○ Вход в аккаунт	62
○ Чат	63
▪ Первоначальная страница:	63
▪ Поиск собеседника	63
▪ Добавление нужного пользователя в друзья.....	63
▪ Отправка сообщения	64
▪ Ответ на сообщение	65
▪ Редактирование сообщения	65
▪ Удаление сообщения.....	67
▪ Выход из аккаунта.....	68
○ В это время в БД.....	68
Общие замечания	70
Выводы	70
Библиография	70

Введение

Практика в компании Cedacri International представляет собой возможность получить практический опыт работы в области различных технологий и инструментов, связанных с разработкой программного обеспечения. Целью практики является приобретение знаний и навыков в разработке программного обеспечения, а также применение лучших практик и методологий веб-разработки.

Описание практики

В ходе практики я работал над следующими заданиями:

1. Калькулятор:

- Реализация стека для оценки арифметических выражений.
- Тестирование программы с использованием составных выражений с операторами и скобками.

2. Создание проекта с использованием Maven:

- Добавление зависимостей в проект.
- Создание многомодульного проекта с зависимостями между модулями.
- Создание поставляемого Jar-файла.

3. Решение упражнений и тестов с использованием SQL на сайте sqlzoo.net.

- Предоставление скриншотов полученных результатов.

4. Jasper Reports:

- Создание отчета в Jasper Studio с использованием таблиц и диапазонов отчета.
- Заполнение отчета данными из различных источников данных.
- Создание DynamicReport и сводного отчета.

5. Junit:

- Использование JUnit для тестирования методов.
- Применение TDD для написания тестируемого кода.
- Проверка покрытия кода для критических частей.

6. Clean Code + Refactoring:

- Практика написания чистого кода в соответствии с принципами, описанными в книге Роберта Мартина.
- Применение рефакторинга для улучшения качества кода.
- Обеспечение правильного форматирования кода и соблюдения руководства по стилю Java.

7. Hibernate:

- Настройка соединения Hibernate.
- Создание отображений домена с использованием аннотаций.
- Создание схемы базы данных на основе отображений.
- Реализация обобщенного объекта доступа к данным (Generic DAO) и тестирование CRUD-операций.

8. Spring:

- Создание приложения Spring с использованием конфигурации аннотациями.
- Подключение к базе данных в приложении.
- Настройка Hibernate в приложении.
- Создание приложения.
- Создание веб приложения

Просмотр приложений:

- Приложение **Web Chat** доступно по ссылке: <https://github.com/n1kry/chat-app>
- Приложение **калькулятор** доступно по ссылке: <https://github.com/n1kry/calculator>
- Приложение **Jasper Reports** доступно по ссылке: <https://github.com/n1kry/JasperReports>
- Приложение **Junit** доступно по ссылке: <https://github.com/n1kry/junit>
- Приложение **Hibernate** доступно по ссылке: <https://github.com/n1kry/hibernate>
- Приложение **Spring** доступно по ссылке: <https://github.com/n1kry/spring>

Содержание действий и задания

1. Data Structures Calculator

- Продумывание работы алгоритма

(Для реализации данного алгоритма было принято решение форматировать входное выражение, в понятное ему, для этого использовался метод `formattingString(String exp)` возвращающий массив символов)

```
public static char[] formattingString(String expression) {
    String formatString = expression.replaceAll("\\s", "");
    formatString = formatString.replaceAll("\\\\", "\\");

    Pattern pattern = Pattern.compile("[+*/()-\\\\()"];
    Matcher matcher = pattern.matcher(formatString);

    while (matcher.find()) {
        formatString = formatString.replace(matcher.group(),
            matcher.group().charAt(0) + "-1*(");
    }
    pattern = Pattern.compile("[0-9]\\\\()");
    matcher = pattern.matcher(formatString);

    while (matcher.find()) {
        formatString = formatString.replace(matcher.group(),
            matcher.group().charAt(0) + "-1*(");
    }
}
```

```

matcher.group().charAt(0)+"*(";
    }

    formatString = formatString.replaceAll("^-\\(", "-1*(");

    long openBrackets = formatString.chars().filter(ch -> ch ==
'(').count();
    long closeBrackets = formatString.chars().filter(ch -> ch ==
')').count();
    long newSize = openBrackets - closeBrackets;

    char[] tokens = new char[(int) (formatString.length() + newSize)];

System.arraycopy(formatString.toCharArray(),0,tokens,0,formatString.len
gth());

    for (int i = formatString.length(); i < tokens.length; i++) {
        tokens[i] = ')';
    }

    return tokens;
}

```

Выражение на выходе: $-1*(2*23)+(22-1-(1+1))*(1+4)$

- Сам алгоритм:

```

public static int evaluate(String expression) {

    char[] tokens = formattingString(expression);
    System.out.println(tokens);
    boolean negativePermission = true;

    // Stack for operands
    Stack<Integer> values = new Stack<>();

    // Stack for Operators
    Stack<Character> ops = new Stack<>();

    try {
        for (int i = 0; i < tokens.length; i++) {
            // Push operands onto the stack
            if (tokens[i] >= '0' && tokens[i] <= '9' ||
negativePermission && tokens[i] == '-') {
                StringBuilder sb = new StringBuilder();
                if (tokens[i] == '-') {
                    sb.append(tokens[i++]);
                }
                while (i < tokens.length && tokens[i] >= '0' &&
tokens[i] <= '9') {
                    sb.append(tokens[i++]);
                }
                values.push(Integer.parseInt(sb.toString()));
                negativePermission = false;
                i--;
            }

            // Handle opening parentheses
            else if (tokens[i] == '(') {
                ops.push(tokens[i]);
            }
        }
    }
}

```

```

    }

    // Handle closing parentheses
    else if (tokens[i] == ')') {
        while (!ops.empty() && ops.peek() != '(') {
            values.push(applyOp(ops.pop(), values.pop(),
values.pop()));
        }
        if (!ops.empty()) {
            ops.pop(); // remove opening parenthesis from stack
        }
    }

    // Handle operators
    else if (tokens[i] == '+' || tokens[i] == '-' ||
tokens[i] == '*' || tokens[i] == '/') {
        while (!ops.empty() && hasPrecedence(tokens[i],
ops.peek())) {
            values.push(applyOp(ops.pop(), values.pop(),
values.pop()));
        }
        ops.push(tokens[i]);
        negativePermission = true;
    }
}

// Evaluate remaining expressions
while (!ops.empty()) {
    values.push(applyOp(ops.pop(), values.pop(),
values.pop()));
}

} catch (EmptyStackException e) {
    throw new RuntimeException(new Throwable("Wrong input
expression"));
}
// Final result is the only element on the stack
return values.pop();
}

// Apply the operator to the two operands
public static int applyOp(char op, int b, int a) {
    switch (op) {
        case '+' -> {
            return a + b;
        }
        case '-' -> {
            return a - b;
        }
        case '*' -> {
            return a * b;
        }
        case '/' -> {
            if (b == 0)
                throw new UnsupportedOperationException("Cannot divide
by zero");
            return a / b;
        }
    }
    return 0;
}

// Check operator precedence

```



```
public static boolean hasPrecedence(char op1, char op2) {
    if (op2 == '(' || op2 == ')')
        return false;
    return (op1 != '*' && op1 != '/') || (op2 != '+' && op2 != '-');
}
```

Вывод:

```
C:\Users\crme029\Desktop\jdk-17.0.7\bin\java.exe
-(2*23)+(22-1-(1+1))(1+4)
-1*(2*23)+(22-1-(1+1))*(1+4)
49
```

Рисунок 1. Вывод решения программы «Калькулятор»

2. Maven

- Разбиение калькулятора на классы и модули:

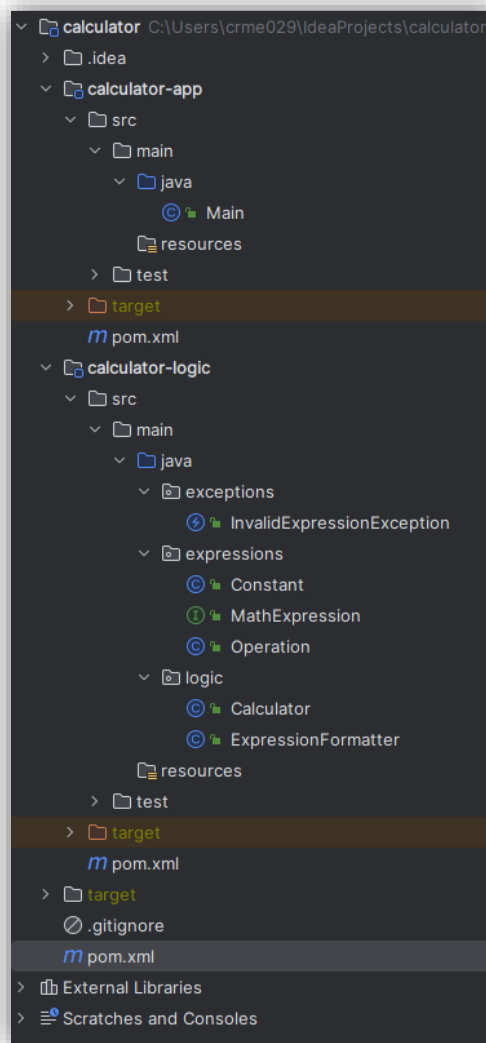


Рисунок 2. Структура приложения Maven

- Добавление зависимостей для модулей и самих модулей
 - parent pom.xml:

```
<groupId>org.example</groupId>
<artifactId>calculator</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>pom</packaging>
<modules>
  <module>calculator-logic</module>
  <module>calculator-app</module>
</modules>
```

- app pom.xml:

```
<dependencies>
  <dependency>
    <groupId>org.example</groupId>
    <artifactId>calculator-logic</artifactId>
    <version>1.0-SNAPSHOT</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

3. SQL

- Прохождение задач и тестов на сайте sqlzoo.net:

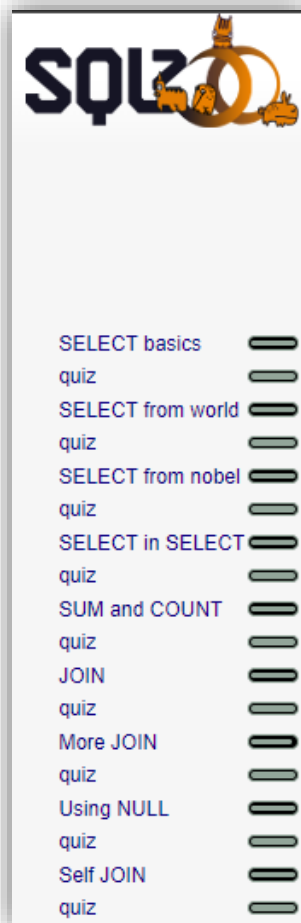


Рисунок 3. Скриншот выполненных заданий SQLZoo

4. Jasper Reports

- Созданный отчет в графическом редакторе Jasper Studio:

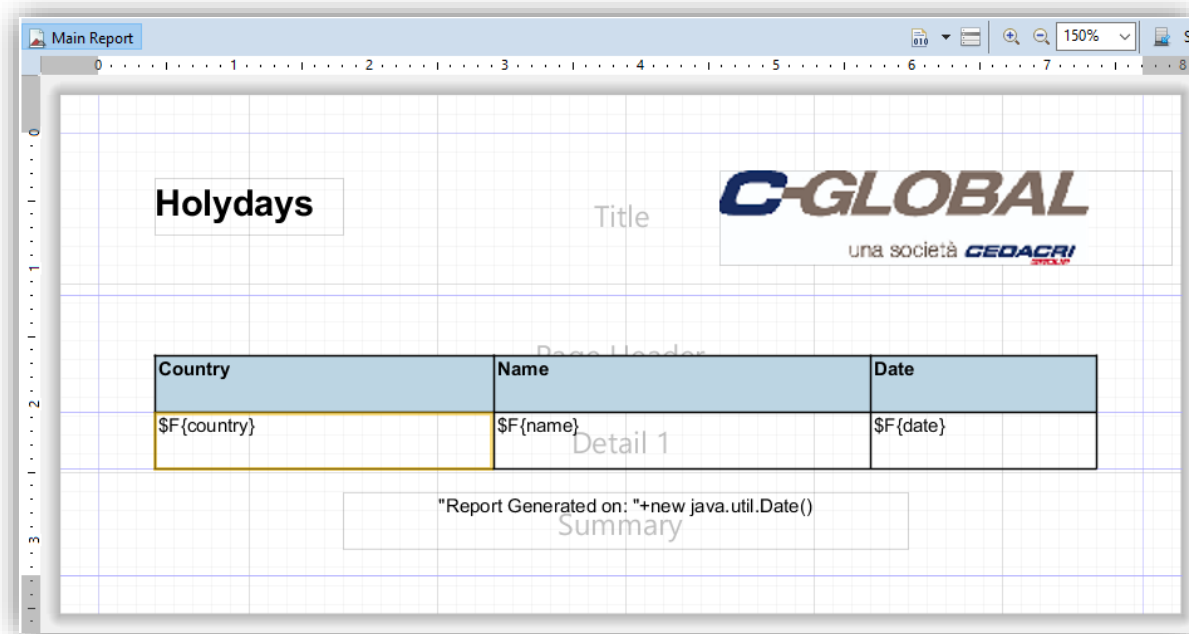


Рисунок 4. Шаблон репорта составленный в Jasper Studio

Country	Name	Date
Italia	Capodanno	1/1/21, 12:00 AM
Italia	Epifania	1/6/21, 12:00 AM
Italia	Pasqua	4/4/21, 12:00 AM
Italia	Pasquetta(Lunedì di Pasqua)	4/5/21, 12:00 AM
Italia	Festa della Liberazione d'Italia	4/25/21, 12:00 AM
Italia	Festa del lavoro o festa dei lavoratori	5/1/21, 12:00 AM
Italia	Festa della Repubblica italiana	6/2/21, 12:00 AM
Italia	Festa dell'Assunzione/Ferragosto	8/15/21, 12:00 AM
Italia	Festa di Ognissanti	11/1/21, 12:00 AM
Italia	Festa dell'Immacolata Concezione	12/8/21, 12:00 AM

Рисунок 5. Вывод репорта составленный в Jasper Studio

- Использование скомпилированного файла jrxml из задания ранее создание репортов использующие данные предоставленные в Java коде. Здесь представлен код сервиса, который преобразует данные полученные из БД в нужные нам для последующих реализаций:

```
public class HolidayService {
    private final IHolidayDao dao = new HolidayDao();

    public List<Map<String,?>> findAllToListMaps() throws SQLException
    {
        ResultSet resultSet = dao.findAll();
        List<Map<String,?>> maps = new ArrayList<>();
        while(resultSet.next()) {
            Map<String,Object> map = new HashMap<>();
            map.put(COUNTRY_FIELD,
resultSet.getString(COUNTRY_FIELD));
            map.put(DATE_FIELD, resultSet.getDate(DATE_FIELD));
            map.put(NAME_FIELD, resultSet.getString(NAME_FIELD));
            maps.add(map);
        }
        return maps;
    }

    public List<Holiday> findAllToListBeans() throws SQLException {
        ResultSet resultSet = dao.findAll();
```

```

        List<Holiday> holidays = new ArrayList<>();
        while (resultSet.next()) {
            Holiday holiday = new Holiday();
            holiday.setCountry(resultSet.getString(COUNTRY_FIELD));
            holiday.setDate(resultSet.getDate( DATE_FIELD));
            holiday.setName(resultSet.getString( NAME_FIELD));
            holidays.add(holiday);
        }
        return holidays;
    }

    public ResultSet findAll() {
        return dao.findAll();
    }

    public ResultSet findAllByMonths() {
        return dao.findAllByMonths();
    }

    public ResultSet countAllHolidaysPerMonth() {
        return dao.countAllHolidaysPerMonth();
    }
}

```

○ JRMapCollectionDataSource:

```

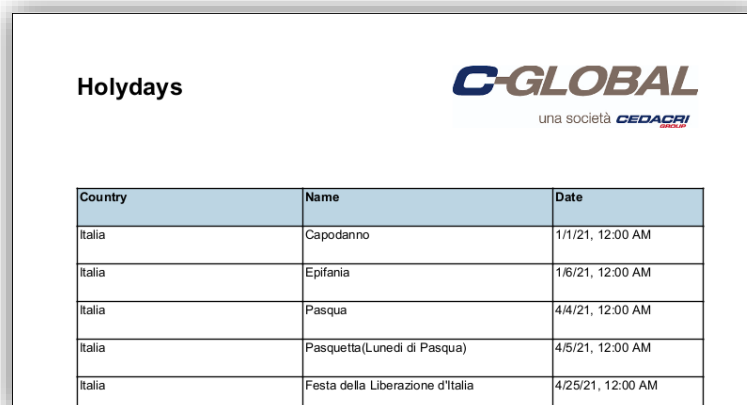
@Override
public void start() {
    try {
        List<Map<String,?>> maps = new
        ArrayList<>(service.findAllToListMaps());

        JRMapCollectionDataSource dataSource = new
        JRMapCollectionDataSource(maps);

        Reportable.showReportFromDataSource(dataSource);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Результат:



Country	Name	Date
Italia	Capodanno	1/1/21, 12:00 AM
Italia	Epifania	1/6/21, 12:00 AM
Italia	Pasqua	4/4/21, 12:00 AM
Italia	Pasquetta(Lunedì di Pasqua)	4/5/21, 12:00 AM
Italia	Festa della Liberazione d'Italia	4/25/21, 12:00 AM

Рисунок 6. Вывод репорта составленного в Jasper Studio, но заполненного в Java с помощью JRMapDataSource

○ JRBeanCollectionDataSource:

```

@Override
public void start() {
    try {
        List<Holiday> holidays = service.findAllToListBeans();

        JRBeanCollectionDataSource dataSource = new
        JRBeanCollectionDataSource(holidays);

        Reportable.showReportFromDataSource(dataSource);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Результат:



Country	Name	Date
Italia	Capodanno	1/1/21, 12:00 AM
Italia	Epifania	1/6/21, 12:00 AM
Italia	Pasqua	4/4/21, 12:00 AM
Italia	Pasquetta (Lunedì di Pasqua)	4/5/21, 12:00 AM
Italia	Festa della Liberazione d'Italia	4/25/21, 12:00 AM

Рисунок 7. Вывод репорта составленного в Jasper Studio, но заполненного в Java с помощью JRBeanDataSource

- JRResultSetCollectionDataSource:

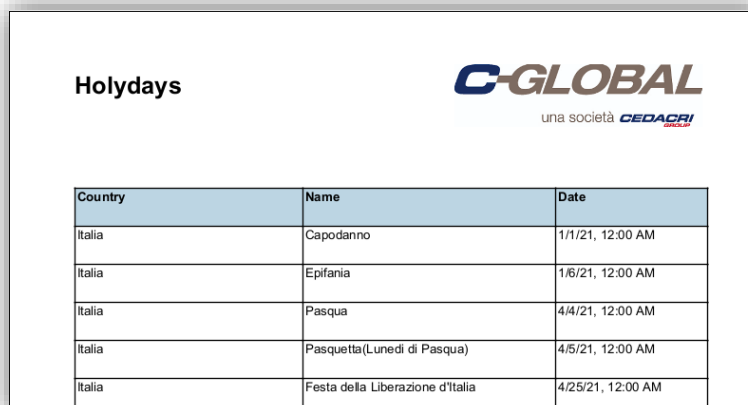
```

@Override
public void start() {
    JRResultSetDataSource resultSetDataSource = new
    JRResultSetDataSource(service.findAll());

    Reportable.showReportFromDataSource(resultSetDataSource);
}

```

Результат:



Country	Name	Date
Italia	Capodanno	1/1/21, 12:00 AM
Italia	Epifania	1/6/21, 12:00 AM
Italia	Pasqua	4/4/21, 12:00 AM
Italia	Pasquetta (Lunedì di Pasqua)	4/5/21, 12:00 AM
Italia	Festa della Liberazione d'Italia	4/25/21, 12:00 AM

- Dynamic Report

(Создание репорта с нуля при помощи шаблона строителя *DinamicReports* из библиотеки *dynamicreports*)

```
@Override
public void start() {
    JRResultSetDataSource resultSetDataSource = new
    JRResultSetDataSource(service.findAll());
    try {
        report()
            .setDataSource(resultSetDataSource)
            .pageHeader(
                cmp.horizontalList(

cmp.text("Country").setStyle(BOLD_BORDERED),

cmp.text("Name").setStyle(BOLD_BORDERED),

cmp.text("Date").setStyle(BOLD_BORDERED)

).setHeight(30).setStyle(HEADER_STYLE)
            )
            .fields(field(COUNTRY_FIELD,
type.stringType()), field(
DATE_FIELD, type.dateType()),
field(NAME_FIELD, type.stringType()))
            .detail(
                cmp.horizontalList(

cmp.text(exp.jasperSyntax("$F{" + COUNTRY_FIELD + "}")).setStyle(
DETAIL_BORDERED),

cmp.text(exp.jasperSyntax("$F{" + NAME_FIELD + "}")).setMinHeight(30)
.setStyle(DETAIL_BORDERED),

cmp.text(exp.jasperSyntax("$F{" + DATE_FIELD + "}")).setStyle(DETAIL_
BORDERED)

                )
            )
            .title(cmp.horizontalList(cmp.text("Holiday").setStyle(TITLE_STYL
E), cmp.image(JasperConsts.IMAGE_PATH)))
            .pageFooter(cmp.pageXofY())
            .show();
    } catch (DRException e) {
        throw new RuntimeException(e);
    }
}
```

Результат:

Holiday		
 una società 		
Country	Name	Date
Italia	Capodanno	1/1/21, 12:00 AM
Italia	Epifania	1/6/21, 12:00 AM
Italia	Pasqua	4/4/21, 12:00 AM
Italia	Pasquetta(Lunedì di Pasqua)	4/5/21, 12:00 AM
Italia	Festa della Liberazione d'Italia	4/25/21, 12:00 AM

Рисунок 9. Вывод репорта составленного в Java с помощью Dynamic Reports

○ Crosstab Report

(Создание графиков и итогов с помощью шаблона строителя как с DynamicReports):

```
@Override
public void start() {
    JRResultSetDataSource resultSetDataSource = new
    JRResultSetDataSource(service.findAllByMonths());
    JRResultSetDataSource resultSetDataSourceChart = new
    JRResultSetDataSource(service.countAllHolidaysPerMonth());

    TextColumnBuilder<Integer> monthColumn =
    col.column(MONTH_FIELD, DATE_FIELD, type.integerType());

    try {
        report()

        .title(cmp.horizontalList(cmp.text("Holiday").setStyle(TITLE_STYL
        E), cmp.image(IMAGE_PATH)))

        .setTitleStyle(stl.style().setLeftPadding(20).setRightPadding(20)
        .setTopPadding(20))
        .summary(
            ctab.crosstab()

        .headerCell(cmp.text("State/Messe").setStyle(BOLD_HEADER_CELL_BOR
        DERED))

        .addRowGroup(ctab.rowGroup(COUNTRY_FIELD,
        String.class).setTotalHeaderStyle(TOTAL_BORDERED))

        .addColumnGroup(ctab.columnGroup(monthColumn).setTotalHeaderStyle
        (TOTAL_BORDERED_RIGHT))

        .addMeasure(ctab.measure(ID_FIELD, Integer.class,
        Calculation.COUNT))

            .setCellWidth(30)

        .setStyle(stl.style().setBottomPadding(40))
            .setCellStyle(CELLS_BORDERED)

        .setGrandTotalStyle(GRAND_TOTAL_BORDERED)
            .setGroupStyle(HEADER_BORDERED),
```



```

        cht.barChart()
            .setTitle("Holidays per month:")
        .setCategory(col.column(MONTH_FIELD, String.class))
            .series(
                cht.serie(col.column("Holidays count", COUNT_FIELD,
                    Integer.class))
                .setSeries(col.column("Countries", COUNTRY_FIELD, String.class))
            )
        .setDataSource(resultSetDataSourceChart)
            )
            .setSummaryStyle(stl.style().setPadding(20))
            .setDataSource(resultSetDataSource)
            .show();
    } catch (DREException e) {
        throw new RuntimeException(e);
    }
}

```

Результат:

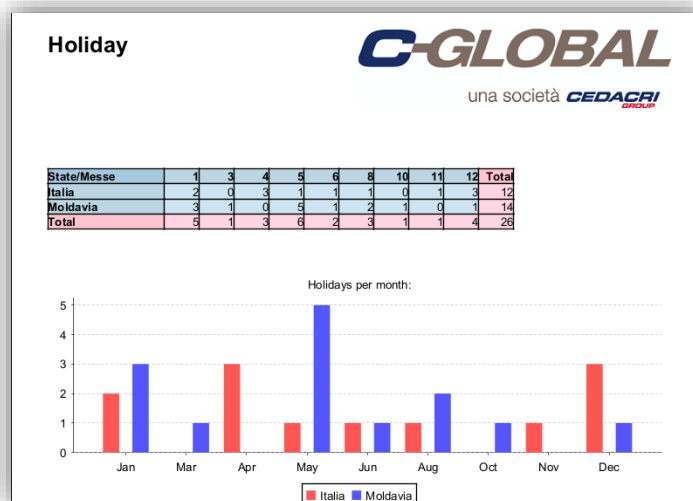


Рисунок 10. Вывод графика и гистограммы составленного в Java с помощью Dynamic Reports

5. Junit

- Создание домена

(В моем случае это парковка автомобилей, где есть список авто на парковке и учет его активности на данный момент. Выбор сделан с целью создания множества связанных и сложных методов, которые можно будет протестировать, используя Junit)

- Junit тесты для моего приложения:

```

@DisplayName("model.Car parking")
class CarParkingTest {

    private CarParking carParking;
    private Car hondaCivicOld;
    private Car hondaCivicNew;
    private Car bmwM5New;
    private Car bmwM5Old;
    private Car mercedesE220Old;
    private Car mercedesE220New;

    @BeforeEach
    void init() {
        carParking = new CarParking();
        hondaCivicOld = new Car("Civic", "Honda", "ADD231", 2006);
        hondaCivicNew = new Car("Civic", "Honda", "ASW001", 2020);
        bmwM5New = new Car("M5", "BMW", "EPP1", 2016);
        bmwM5Old = new Car("M5", "BMW", "EPJ112", 2006);
        mercedesE220Old = new Car("E220", "Mercedes", "GTF241", 2008);
        mercedesE220New = new Car("E220", "Mercedes", "GHR098", 2015);
    }

    @Nested
    @DisplayName("Is empty")
    class IsEmpty {
        @Test
        @DisplayName("when no cars is added to it")
        public void emptyCarParkingWhenNoCarAdded() {
            List<Car> cars = carParking.cars();
            assertTrue(cars.isEmpty(), "model.CarParking should be
empty");
        }

        @Test
        @DisplayName("when add is called without cars")
        public void emptyCarParkingWhenAddIsCalledWithoutCars() {
            carParking.add();
            List<Car> cars = carParking.cars();
            assertTrue(cars.isEmpty(), "model.CarParking should be
empty");
        }
    }

    @Nested
    @DisplayName("after adding cars")
    class CarsAreAdded {
        @Test
        @DisplayName("contains two cars")
        void carParkingContainsTwoCarsWhenTwoCarsAdded() {
            carParking.add(hondaCivicNew, bmwM5New);
            List<Car> cars = carParking.cars();
            assertEquals(2, cars.size(), "model.CarParking should have
two cars");
        }

        @Test
        @DisplayName("returns an immutable cars collection to client")
        void carsParkingIsImmutableForClients() {
            carParking.add(hondaCivicNew, hondaCivicOld);
            List<Car> cars = carParking.cars();
            try {
                cars.add(bmwM5New);
                fail("Should not be able to add book to books");
            } catch (Exception e) {

```

```

        assertTrue(e instanceof UnsupportedOperationException,
"CarsParking should throw UnsupportedOperationException");
    }
}

@Test
@DisplayName("throws exception when out of capacity")
void throwsExceptionWhenCarsAreAddedAfterCapacityIsReached() {
    CarParking parking = new CarParking(3);
    parking.add(hondaCivicOld, hondaCivicNew, mercedesE220New);
    CarParkingCapacityReached throwException =
assertThrows(CarParkingCapacityReached.class, () ->
parking.add(bmwM5New));
    assertEquals("Out of capacity",
throwException.getMessage());
}

@Nested
@DisplayName("Is arranged")
class WhewArranged {
    @Test
    @DisplayName("lexicographically by model")
    void carParkingArrangedByCarMaker() {
        carParking.add(hondaCivicNew, bmwM5New, mercedesE220New);
        List<Car> cars = carParking.arrange();
        assertEquals(Arrays.asList(hondaCivicNew,mercedesE220New,
bmwM5New), cars, "Cars in carsParking should be arranged by car
model");
    }

    @Test
    @DisplayName("by user provided option (lexicographically by
model DESC)")
    void carParkingArrangedByCarMakerByUserProvidedOption() {
        carParking.add(hondaCivicNew, bmwM5New, mercedesE220New);
        List<Car> cars =
carParking.arrange(Comparator.<Car>naturalOrder().reversed());

        assertEquals(Arrays.asList(bmwM5New,mercedesE220New,hondaCivicNew),cars
,"Cars in carsParking should be arranged in descending order by car
model");
    }

    @Test
    @DisplayName("by user provided option (made year in ASC
order)")
    void carParkingArrangedByCarMadeYearByUserProvidedOption() {
        carParking.add(hondaCivicNew, bmwM5New, mercedesE220New,
hondaCivicOld, mercedesE220Old);
        List<Car> cars =
carParking.arrange(Comparator.comparing(Car::getMadeYear));
        assertEquals(Arrays.asList(hondaCivicOld, mercedesE220Old,
mercedesE220New, bmwM5New, hondaCivicNew), cars, "Cars in carParking
are arranged by car made year in ascending order");
    }
}

@Nested
@DisplayName("cars are grouped by")
class GroupBy {
    @Test
    @DisplayName("maker")

```

```

        void groupCarsInCarParkingByMaker() {
            carParking.add(hondaCivicNew, bmwM5New, mercedesE220New,
hondaCivicOld, mercedesE220Old, bmwM5Old);
            Map<String, List<Car>> cars = carParking.groupByMaker();

assertTrue(cars.get("Honda").containsAll(Arrays.asList(hondaCivicNew,
hondaCivicOld)), "Cars grouped by Honda");

assertTrue(cars.get("BMW").containsAll(Arrays.asList(bmwM5Old,
bmwM5New)), "Cars grouped by BMW");

assertTrue(cars.get("Mercedes").containsAll(Arrays.asList(mercedesE220N
ew, mercedesE220Old)), "Cars grouped by Mercedes");
        }

@Test
@DisplayName("year")
void groupCarsInCarParkByMadeYear() {
    carParking.add(hondaCivicNew, bmwM5New, mercedesE220New,
hondaCivicOld, mercedesE220Old, bmwM5Old);
    Map<Integer, List<Car>> cars =
carParking.groupBy((Car::getMadeYear));

assertTrue(cars.get(2006).containsAll(Arrays.asList(bmwM5Old,
hondaCivicOld)), "Cars grouped by 2006");
    assertTrue(cars.get(2020).contains(hondaCivicNew), "Cars
grouped by 2020");
    assertTrue(cars.get(2016).contains(bmwM5New), "Cars grouped
by 2016");
    assertTrue(cars.get(2015).contains(mercedesE220New), "Cars
grouped by 2015");
}

}

@Nested
@DisplayName("Find by")
class FindBy {
    @Test
    @DisplayName("maker")
    void findCarsInCarParkByMaker() {
        carParking.add(hondaCivicNew, bmwM5New, mercedesE220New,
hondaCivicOld, mercedesE220Old, bmwM5Old);
        List<Car> cars =
carParking.findCarsByMaker("Honda".toLowerCase());
        assertEquals(Arrays.asList(hondaCivicNew, hondaCivicOld),
cars, "Cars in carParking should be made by Honda");
    }
}

@Nested
@DisplayName("statistic")
class Stats {
    @Test
    @DisplayName("how many cars in rent now")
    void printCurrentStatisticWithAddedCars() {
        Car car = new Car("Passat", "VolksWagen", "SVGT245", 1992);
        car.setStartedRentOn(LocalDate.now());
        carParking.add(hondaCivicNew, bmwM5New, mercedesE220New,
bmwM5Old, car);
        assertEquals(20, carParking.statistic(), "Percentage of
rented cars");
    }
}
@Test

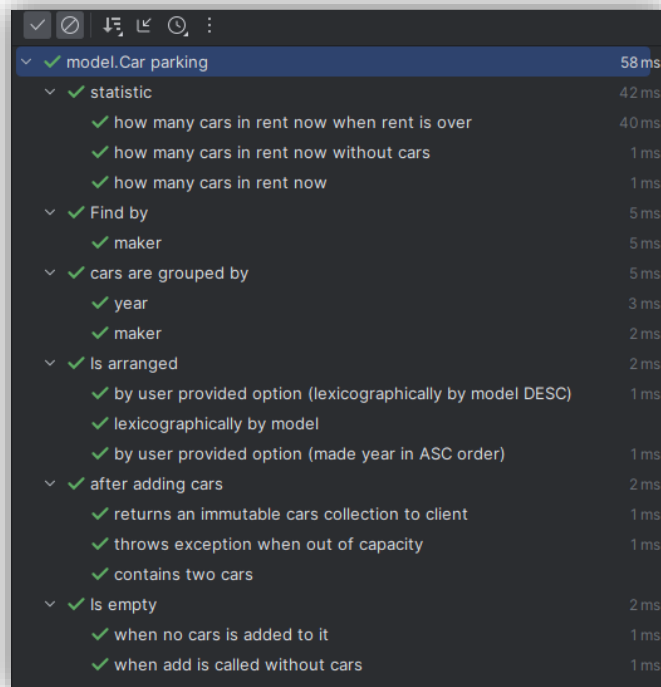
```

```

        @DisplayName("how many cars in rent now without cars")
        void printCurrentStatisticWithoutCars() {
            assertEquals(0, carParking.statistic(), "Percentage of
rented cars");
        }
        @Test
        @DisplayName("how many cars in rent now when rent is over")
        void printCurrentStatisticWhenRentOver() {
            Car car = new Car("Passat", "VolksWagen", "SVGT245", 1992);
            car.setFinishedRentOn(LocalDate.now());
            carParking.add(hondaCivicNew, bmwM5New, mercedesE220New,
bmwM5Old, car);
            assertEquals(0, carParking.statistic(), "Percentage of
rented cars");
        }
    }
}

```

Результат:



Test Name	Duration
model.Car parking	58 ms
statistic	42 ms
how many cars in rent now when rent is over	40 ms
how many cars in rent now without cars	1 ms
how many cars in rent now	1 ms
Find by	5 ms
maker	5 ms
cars are grouped by	5 ms
year	3 ms
maker	2 ms
Is arranged	2 ms
by user provided option (lexicographically by model DESC)	1 ms
lexicographically by model	1 ms
by user provided option (made year in ASC order)	1 ms
after adding cars	2 ms
returns an immutable cars collection to client	1 ms
throws exception when out of capacity	1 ms
contains two cars	1 ms
Is empty	2 ms
when no cars is added to it	1 ms
when add is called without cars	1 ms

Рисунок 11. Результат тестов приложения «Парковка авто»

- Конвертер для того, чтобы заполнить базу данных используя файл CSV

(Сделать конвертер для того, чтобы заполнить базу данных используя файл CSV формата)

```

public class CSVToDBConverter {
    public static final String CSV_FILE =
"src/main/resources/cars.csv";
    private final CarsDBConnection dbConnection = new
CarsDBConnection();

    private static final String QUERY = "INSERT INTO car ("

```

```

        +"model","
        +"maker","
        +"carNumber","
        +"madeYear","
        +"startedRentOn","
        +"finishedRentOn"
        +"VALUES (?,?,?,?,?,?);";

    public void convert() {
        try (Connection connection = dbConnection.getConnection();
            PreparedStatement pstmt =
connection.prepareStatement(QUERY);
            Statement statement = connection.createStatement()) {

            statement.executeUpdate("DELETE FROM car;");

            CSVReader reader = new CSVReader(new FileReader(CSV_FILE));
            reader.readNext();
            String[] nextLine = reader.readNext();

            do {
                pstmt.setString(1,nextLine[0]);
                pstmt.setString(2,nextLine[1]);
                pstmt.setString(3,nextLine[2]);
                if (Integer.parseInt(nextLine[3]) > 1900) {
                    pstmt.setInt(4, Integer.parseInt(nextLine[3]));
                } else {
                    continue;
                }

                if (!nextLine[4].equals("")) {
                    pstmt.setDate(5, Date.valueOf(nextLine[4]));
                } else {
                    pstmt.setDate(5, null);
                }

                if (!nextLine[5].equals("")) {
                    pstmt.setDate(6, Date.valueOf(nextLine[5]));
                } else {
                    pstmt.setDate(6, null);
                }
                pstmt.executeUpdate();

            } while ((nextLine = reader.readNext()) != null);

            } catch (SQLException e) {
                throw new RuntimeException(e);
            } catch (FileNotFoundException e) {
                throw new RuntimeException(e);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
}

```

Результат:

Входные данные (cars.csv):

```

model,maker,carNumber,madeYear,startedRentOn,finishedRentOn
Corolla,Toyota,AB123CD,2018,,
Civic,Honda,XY789ZY,2017,,
Accord,Honda,AB456CD,2020,,
Model S,Tesla,XY789AB,2021,,
X5,BMW,CD123AB,2015,,
E-Class,Mercedes-Benz,AA123BB,2019,,
Model 3,Tesla,ZZ987YY,2020,,
Escape,Ford,BC234DE,2018,,
CX-5,Mazda,EE111FF,2017,,
S90,Volvo,DD222GG,2021,,
Polo,Volkswagen,AC456FG,2016,,
Optima,Kia,DD444HH,2019,,
RAV4,Toyota,BB222CC,2017,,
Camry,Toyota,CC333AA,2021,,
Q7,Audi,AA888ZZ,2020,,
Camaro,Chevrolet,AB444CD,2018,,
F-Type,Jaguar,YY111XX,2017,,
Wrangler,Jeep,BC345DE,2021,,
Tucson,Hyundai,EE222FF,2019,,
A4,Audi,AC333BB,1800,,

```

Рисунок 12. CSV файл с автомобилями на парковке

Выходные данные:

	id	model	maker	carnumber	madeyear	startedrenton	finishedrenton
1	233	Corolla	Toyota	AB123CD	2,018	[NULL]	[NULL]
2	234	Civic	Honda	XY789ZY	2,017	[NULL]	[NULL]
3	235	Accord	Honda	AB456CD	2,020	[NULL]	[NULL]
4	236	Model S	Tesla	XY789AB	2,021	[NULL]	[NULL]
5	237	X5	BMW	CD123AB	2,015	[NULL]	[NULL]
6	238	E-Class	Mercedes-Benz	AA123BB	2,019	[NULL]	[NULL]
7	239	Model 3	Tesla	ZZ987YY	2,020	[NULL]	[NULL]
8	240	Escape	Ford	BC234DE	2,018	[NULL]	[NULL]
9	241	CX-5	Mazda	EE111FF	2,017	[NULL]	[NULL]
10	242	S90	Volvo	DD222GG	2,021	[NULL]	[NULL]
11	243	Polo	Volkswagen	AC456FG	2,016	[NULL]	[NULL]
12	244	Optima	Kia	DD444HH	2,019	[NULL]	[NULL]
13	245	RAV4	Toyota	BB222CC	2,017	[NULL]	[NULL]
14	246	Camry	Toyota	CC333AA	2,021	[NULL]	[NULL]
15	247	Q7	Audi	AA888ZZ	2,020	[NULL]	[NULL]
16	248	Camaro	Chevrolet	AB444CD	2,018	[NULL]	[NULL]
17	249	F-Type	Jaguar	YY111XX	2,017	[NULL]	[NULL]
18	250	Wrangler	Jeep	BC345DE	2,021	[NULL]	[NULL]
19	251	Tucson	Hyundai	EE222FF	2,019	[NULL]	[NULL]

Рисунок 13. Таблица в базе данных после добавления данных из CSV файла

6. Clean Code + Refactoring

Цель: писать чистый код по книгам Мартина. Соблюдать руководство по стилю Java. Использовать рефакторинг.

7. Hibernate

(Приложение для просмотра книг и их авторов (у одной книги может быть много авторов, как и у одного автора может быть много книг))

- Конфигурация Hibernate для подключения к базе данных:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="connection.url">jdbc:postgresql://localhost:5433/postgres</proper
ty>
        <property name="connection.username">postgres</property>
        <property name="connection.password">1212</property>
        <property
name="hibernate.connection.characterEncoding">utf8</property>

        <property name="show_sql">true</property>
        <property name="format_sql">true</property>

        <property
name="hibernate.current_session_context_class">thread</property>
        <property name="hbm2ddl.auto">create-drop</property>

        <mapping class="model.Author"/>
        <mapping class="model.Book"/>
    </session-factory>
</hibernate-configuration>
```

- Создания моделей, соответствующих таблицам из БД:

- Author (Модель):

```
@Entity
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
@Builder
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @ToString.Exclude
    private Integer id;

    @ToString.Include
    private String name;

    @ToString.Exclude
    @ManyToMany(mappedBy = "authorId", cascade = CascadeType.ALL)
    private List<Book> books;
}
```

- Book (Модель):

```
@Entity
@Getter
@Setter
@ToString
@NoArgsConstructor
@AllArgsConstructor
```



```
@Builder
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @ToString.Exclude
    private Integer id;

    @ToString.Include
    private String name;

    @ToString.Exclude
    @ManyToMany
    @JoinColumn(name = "id_author")
    private List<Author> authorId;
}
```

- Создание репозитория (DAO) для доступа к CRUD операциям с БД:

- AuthorDAO:

```
public interface IAuthorDao<T extends Author, Id extends Integer>
{
    List<T> findAll();

    T findById(Id id);

    void update(T updated);

    List<String> findAuthorNameByBookName(String bookName);

    Map<T, List<Book>> getAuthorsBooksMap();

    void save(T author);

    void delete(T author);
}
```

- BookDAO:

```
public interface IBookDao<T extends Book, Id extends Integer> {
    List<T> findAll();

    T findById(Id id);

    void update(T updated);

    void save(T book);

    void delete(T book);
}
```

- Создание тестов для проверки работоспособности всех методов из сервисов:

- Author Service тесты:

```
class AuthorServiceTest {

    static AuthorService authorService = new AuthorService();
    static BookService bookService = new BookService();

    @BeforeAll
    static void setup() {
        authorService.save(Author.builder().name("John").build());
    }
}
```

```

        authorService.save(Author.builder().name("John
Second").build());
        authorService.save(Author.builder().name("John
Third").build());

        bookService.save(Book.builder().name("Marts
Journey").authorId(List.of(Author.builder().id(1).build(), Author.
builder().id(2).build())).build());

bookService.save(Book.builder().name("Scam").authorId(List.of(Aut
hor.builder().id(2).build(), Author.builder().id(3).build())).buil
d());
    }

    @Test
    void findById() {
        // Arrange
        Integer id = 3;

        // Act
        Author author = authorService.findById(id);

        // Assert
        assertNotNull(author);
        assertEquals(Optional.of(id).get(),
Optional.of(author.getId()).get().intValue());
    }

    @Test
    void findAll() {
        // Arrange
        AuthorService authorService = new AuthorService();

        // Act
        List<Author> authors = authorService.findAll();

        // Assert
        assertNotNull(authors);
        assertFalse(authors.isEmpty());
    }

    @Test
    void update() {
        // Arrange
        Author author = new Author();
        author.setId(3);
        author.setName("John Doe");

        // Act
        authorService.update(author);
        Author updatedAuthor =
authorService.findById(author.getId());

        // Assert
        assertEquals(author.getName(), updatedAuthor.getName());
    }

    @Test
    void getAuthorByBookName_ExistingBook() {
        // Arrange
        String bookName = "Scam";

        // Act

```

```

        List<String> authors =
authorService.getAuthorByBookName(bookName);

        // Assert
        assertNotNull(authors);
        assertFalse(authors.isEmpty());
        assertEquals("John Doe", authors.get(0));
    }

    @Test
    public void getAuthorByBookName_NonExistingBook() {
        // Arrange
        String bookName = "Non-existing Book Title";

        // Act
        List<String> authors =
authorService.getAuthorByBookName(bookName);

        // Assert
        assertNotNull(authors);
        assertEquals(Collections.emptyList(), authors);
    }

    @Test
    void getAuthorsBooksMap() {
        // Act
        Map<Author, List<Book>> authorListMap =
authorService.getAuthorsBooksMap();

        // Assert
        assertNotNull(authorListMap);
        assertFalse(authorListMap.isEmpty());
    }

    @Test
    void save() {
        AuthorService authorService = new AuthorService();
        Author author = new Author();
        author.setName("Jane Doe");

        // Act
        authorService.save(author);
        Author savedAuthor =
authorService.findById(author.getId());

        // Assert
        assertEquals(author.getName(), savedAuthor.getName());
    }

    @Test
    void delete() {
        // Arrange
        AuthorService authorService = new AuthorService();
        Author author = new Author();
        author.setName("John Doe");
        authorService.save(author);

        // Act
        authorService.delete(author);
        Author deletedAuthor =
authorService.findById(author.getId());

        // Assert

```

```

        assertNull(deletedAuthor);
    }
}

```

Результат:

✓ <default package>	366 ms
✓ AuthorServiceTest	366 ms
✓ findById()	25 ms
✓ findAll()	267 ms
✓ update()	8 ms
✓ getAuthorByBookName_ExistingBook()	30 ms
✓ getAuthorByBookName_NonExistingBook()	3 ms
✓ getAuthorsBooksMap()	8 ms
✓ save()	5 ms
✓ delete()	20 ms

Рисунок 14. Результаты тестов сервиса Author и его методов

o Book Service тесты:

```

class BookServiceTest {
    static AuthorService authorService = new AuthorService();
    static BookService bookService = new BookService();

    @BeforeAll
    static void setup() {

        authorService.save(Author.builder().name("John").build());
        authorService.save(Author.builder().name("John
Second").build());
        authorService.save(Author.builder().name("John
Third").build());

        bookService.save(Book.builder().name("Marts
Journey").authorId(List.of(Author.builder().id(1).build(), Author.
builder().id(2).build())).build());

        bookService.save(Book.builder().name("Scam").authorId(List.of(Aut
hor.builder().id(2).build(), Author.builder().id(3).build()))).buil
d());
    }

    @Test
    public void testFindById() {
        // Arrange
        BookService bookService = new BookService();
        Integer id = 1;

        // Act
        Book book = bookService.findById(id);

        // Assert
    }
}

```

```

        assertNotNull(book);
        assertEquals(id, book.getId());
    }

    @Test
    public void testFindAll() {
        // Arrange
        BookService bookService = new BookService();

        // Act
        List<Book> books = bookService.findAll();

        // Assert
        assertNotNull(books);
        assertFalse(books.isEmpty());
    }

    @Test
    public void testUpdate() {
        // Arrange
        BookService bookService = new BookService();
        Book book = new Book();
        book.setId(1);
        book.setName("Updated Title");

        // Act
        bookService.update(book);
        Book updatedBook = bookService.findById(book.getId());

        // Assert
        assertEquals(book.getName(), updatedBook.getName());
    }

    @Test
    public void testSave() {
        // Arrange
        BookService bookService = new BookService();
        Book book = new Book();
        book.setName("New Book");

book.setAuthorId(List.of(Author.builder().id(2).build()));

        // Act
        bookService.save(book);
        Book savedBook = bookService.findById(book.getId());

        // Assert
        assertEquals(book.getName(), savedBook.getName());
    }

    @Test
    public void testDelete() {
        // Arrange
        BookService bookService = new BookService();
        Book book = new Book();
        book.setName("To Be Deleted");

book.setAuthorId(List.of(Author.builder().id(2).build()));
        bookService.save(book);

        // Act
        bookService.delete(book);
        Book deletedBook = bookService.findById(book.getId());
    }

```

```
// Assert
assertNull(deletedBook);
}
}
```

Результат:

✓ BookServiceTest (service)	306 ms
✓ testFindAll()	273 ms
✓ testSave()	7 ms
✓ testFindById()	2 ms
✓ testDelete()	17 ms
✓ testUpdate()	7 ms

Рисунок 15. Результаты тестов сервиса Book и его методов

8. Spring

(На основе задания по Hibernate сделать приложение используя Spring Framework)

- Конфигурация Spring

(для доступа к БД и способа работы с ней (application.properties))

```
spring.datasource.url=jdbc:postgresql://localhost:5433/postgres
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.username=postgres
spring.datasource.password=1212

spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.properties.hibernate.format_sql=true
```

- Создание моделей (Entity) для их связи с БД:

- Author:

```
@Entity
@Getter
@Setter
@AllArgsConstructor
@ToString
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @ToString.Exclude
    private Integer id;

    @ToString.Include
    @Column(nullable = false)
    private String name;

    @ToString.Exclude
    @ManyToMany(mappedBy = "author", cascade = CascadeType.ALL,
```

```

fetch = FetchType.EAGER)
private List<Book> books;

public Author() {
    books = new ArrayList<>();
}

public Author(Integer id, String name) {
    this.id = id;
    this.name = name;
    books = new ArrayList<>();
}
}

```

- Book:

```

@Entity
@Getter
@Setter
@ToString
@ToString.Exclude
@AllArgsConstructor
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @ToString.Exclude
    private Integer id;

    @ToString.Include
    private String name;

    @ToString.Exclude
    @ManyToMany
    private List<Author> author;

    public Book() {
        author = new ArrayList<>();
    }

    public Book(Integer id, String name) {
        this.id = id;
        this.name = name;
        author = new ArrayList<>();
    }
}

```

- Создание DAO и наследование JpaRepository от Spring

(Для доступа к данным БД и простого создания запросов)

- AuthorDAO:

```

public interface AuthorDao extends JpaRepository<Author, Integer> {
    List<Author> findByBooksName(String bookName);
}

```

- BookDAO:

```

public interface BookDao extends JpaRepository<Book, Integer> {
}

```

- Добавление сервисов

(Помогают обрабатывать полученные данные из БД)

- AuthorService:

```
@Service
@AllArgsConstructor
public class AuthorService {

    private final AuthorDao authorDao;

    private final BookDao bookDao;

    public Author save(Author author) {
        return authorDao.save(author);
    }

    public void update(Author author) {
        Author currentAuthor =
authorDao.findById(author.getId()).orElse(null);
        if (currentAuthor != null) {
            if(author.getName() != null &&
!author.getName().equals("")) {
                currentAuthor.setName(author.getName());
            }
            authorDao.save(currentAuthor);
        }
    }

    @Transactional
    public void delete(Author author) {
        author = authorDao.findById(author.getId()).orElse(null);
        if (author != null) {
            List<Book> books = author.getBooks();
            for (Book book : books) {
                book.getAuthor().remove(author);
            }
            bookDao.saveAll(books);

            authorDao.delete(author);
        }
    }

    public List<Author> getAllAuthors() {
        return authorDao.findAll();
    }

    public List<Author> findAuthorsByBookName(String bookName) {
        return authorDao.findByBooksName(bookName);
    }

    public Map<Author, List<Book>> getAuthorsWithBooks() {
        List<Author> authors = authorDao.findAll(); // Retrieve
all authors from the database

        Map<Author, List<Book>> authorsWithBooks = new
HashMap<>();

        for (Author author : authors) {
            List<Book> books = author.getBooks(); // Retrieve the
books for each author
            authorsWithBooks.put(author, books); // Add the
author and their books to the map
        }

        return authorsWithBooks;
    }
}
```



```
}  
}
```

- BookService:

```
@Service  
@AllArgsConstructor  
public class BookService {  
  
    private final BookDao bookDao;  
  
    private final AuthorDao authorDao;  
  
    public List<Book> findAll() {  
        return bookDao.findAll();  
    }  
  
    public Book findById(Integer id) {  
        return bookDao.findById(id).orElse(null);  
    }  
  
    public void save(Book book) {  
        bookDao.save(book);  
    }  
  
    public void update(Book book) {  
        Book currentBook =  
bookDao.findById(book.getId()).orElse(null);  
        if (currentBook != null) {  
            if (book.getName() != null) {  
                currentBook.setName(book.getName());  
            }  
            if (!book.getAuthor().isEmpty()) {  
                currentBook.setAuthor(book.getAuthor());  
            }  
            bookDao.save(currentBook);  
        }  
    }  
  
    @Transactional  
    public void delete(Book book) {  
        book = bookDao.findById(book.getId()).orElse(null);  
        if (book != null) {  
            List<Author> authors = book.getAuthor();  
            for (Author author : authors) {  
                author.getBooks().remove(book);  
            }  
            authorDao.saveAll(authors);  
  
            bookDao.delete(book);  
        }  
    }  
}
```

- Написание тестов с помощью технологии Mockito

(Для тестирования методов, не задевая саму БД)

- AuthorService тест:

```
@ExtendWith(MockitoExtension.class)  
public class AuthorServiceTest {
```

```

@Mock
private AuthorDao authorDao;

@Mock
private BookDao bookDao;

@InjectMocks
private AuthorService authorService;

@Test
public void saveAuthor_ShouldSaveAuthor() {
    int authorId = 1;
    Author author = new Author();
    author.setName("John Doe");

    when(authorDao.save(author)).thenReturn(new
Author(authorId, author.getName()));

    Author savedAuthor = authorService.save(author);

    verify(authorDao).save(author);
    assertNotNull(savedAuthor.getId());
}

@Test
public void updateAuthor_ShouldUpdateAuthor() {
    Author existingAuthor = new Author();
    existingAuthor.setId(1);
    existingAuthor.setName("John Doe");

    Author updatedAuthor = new Author();
    updatedAuthor.setId(1);
    updatedAuthor.setName("Jane Smith");

    when(authorDao.findById(updatedAuthor.getId())).thenReturn(Option
al.of(existingAuthor));

    authorService.update(updatedAuthor);

    verify(authorDao).save(existingAuthor);
    assertEquals("Jane Smith", existingAuthor.getName());
}

@Test
public void deleteAuthor_ShouldRemoveAuthorAndBooks() {
    Author author = new Author();
    author.setId(1);
    author.setName("John Doe");

    Book book1 = new Book();
    book1.setId(1);
    book1.setName("Book 1");
    book1.getAuthor().add(author);
    author.getBooks().add(book1);

    Book book2 = new Book();
    book2.setId(2);
    book2.setName("Book 2");
    book2.getAuthor().add(author);
    author.getBooks().add(book2);
}

```

```

when(authorDao.findById(author.getId())) .thenReturn(Optional.of(author));

        authService.delete(author);

        verify(bookDao).saveAll(author.getBooks());
        verify(authorDao).delete(author);
    }

    @Test
    public void getAllAuthors_ShouldReturnAllAuthors() {
        List<Author> authors = new ArrayList<>();
        authors.add(new Author(1, "John Doe"));
        authors.add(new Author(2, "Jane Smith"));

        when(authorDao.findAll()) .thenReturn(authors);

        List<Author> result = authService.getAllAuthors();

        assertEquals(authors, result);
    }

    @Test
    public void findAuthorsByBookName_ShouldReturnAuthorsForBookName() {
        String bookName = "Book 1";
        List<Author> authors = new ArrayList<>();
        authors.add(new Author(1, "John Doe"));
        authors.add(new Author(2, "Jane Smith"));

        when(authorDao.findByBooksName(bookName)) .thenReturn(authors);

        List<Author> result =
        authService.findAuthorsByBookName(bookName);

        assertEquals(authors, result);
    }

    @Test
    public void getAuthorsWithBooks_ShouldReturnMapOfAuthorsWithBooks() {
        List<Author> authors = new ArrayList<>();
        Author author1 = new Author(1, "John Doe");
        Book book1 = new Book(1, "Book 1");
        author1.getBooks().add(book1);
        book1.getAuthor().add(author1);
        authors.add(author1);

        Author author2 = new Author(2, "Jane Smith");
        Book book2 = new Book(2, "Book 2");
        author2.getBooks().add(book2);
        book2.getAuthor().add(author2);
        authors.add(author2);

        when(authorDao.findAll()) .thenReturn(authors);

        Map<Author, List<Book>> result =
        authService.getAuthorsWithBooks();

        assertEquals(2, result.size());
        assertEquals(Collections.singletonList(book1),
        result.get(author1));
    }

```

```

        assertEquals(Collections.singletonList(book2),
result.get(author2));
    }
}

```

Результат:

✓ AuthorServiceTest (com.example.spring.service)	1 sec 651 ms
✓ deleteAuthor_ShouldRemoveAuthorAndBooks()	1 sec 586 ms
✓ getAllAuthors_ShouldReturnAllAuthors()	43 ms
✓ saveAuthor_ShouldSaveAuthor()	13 ms
✓ findAuthorsByBookName_ShouldReturnAuthorsForBookName()	3 ms
✓ updateAuthor_ShouldUpdateAuthor()	4 ms
✓ getAuthorsWithBooks_ShouldReturnMapOfAuthorsWithBooks()	2 ms

Рисунок 16. Результаты тестов сервиса Author и его методов

o BookService тест:

```

@ExtendWith(MockitoExtension.class)
public class BookServiceTest {

    @Mock
    private BookDao bookDao;

    @Mock
    private AuthorDao authorDao;

    @InjectMocks
    private BookService bookService;

    @Test
    public void findAll_ShouldReturnAllBooks() {
        List<Book> books = new ArrayList<>();
        books.add(new Book(1, "Book 1"));
        books.add(new Book(2, "Book 2"));

        when(bookDao.findAll()).thenReturn(books);

        List<Book> result = bookService.findAll();

        assertEquals(books, result);
    }

    @Test
    public void findById_ExistingId_ShouldReturnBook() {
        int bookId = 1;
        Book book = new Book(bookId, "Book 1");

        when(bookDao.findById(bookId)).thenReturn(Optional.of(book));

        Book result = bookService.findById(bookId);

        assertEquals(book, result);
    }
}

```

```

@Test
public void findById_NonExistingId_ShouldReturnNull() {
    int bookId = 1;

    when(bookDao.findById(bookId)).thenReturn(Optional.empty());

    Book result = bookService.findById(bookId);

    assertNull(result);
}

@Test
public void saveBook_ShouldSaveBook() {
    Book book = new Book();
    book.setName("Book 1");

    bookService.save(book);

    verify(bookDao).save(book);
}

@Test
public void updateBook_ShouldUpdateBook() {
    int bookId = 1;
    Book existingBook = new Book(bookId, "Book 1");
    existingBook.setAuthor(new ArrayList<>());

    Book updatedBook = new Book(bookId, "Updated Book");
    updatedBook.getAuthor().add(new Author(1, "John Doe"));

    when(bookDao.findById(bookId)).thenReturn(Optional.of(existingBook));

    bookService.update(updatedBook);

    verify(bookDao).save(existingBook);
    assertEquals("Updated Book", existingBook.getName());
    assertEquals(1, existingBook.getAuthor().size());
    assertEquals("John Doe",
existingBook.getAuthor().get(0).getName());
}

@Test
public void deleteBook_ShouldRemoveBookAndAuthors() {
    int bookId = 1;
    Book book = new Book(bookId, "Book 1");

    Author author1 = new Author(1, "John Doe");
    book.getAuthor().add(author1);
    author1.getBooks().add(book);

    Author author2 = new Author(2, "Jane Smith");
    book.getAuthor().add(author2);
    author2.getBooks().add(book);

    when(bookDao.findById(bookId)).thenReturn(Optional.of(book));

    bookService.delete(book);

```

```

        verify(authorDao).saveAll(book.getAuthor());
        verify(bookDao).delete(book);
    }
}

```

Результат:

Test Method	Execution Time
BookServiceTest (com.example.spring.service)	1 sec 686 ms
findAll_ShouldReturnAllBooks()	1 sec 640 ms
deleteBook_ShouldRemoveBookAndAuthors()	27 ms
findById_ExistingId_ShouldReturnBook()	4 ms
findById_NonExistingId_ShouldReturnNull()	4 ms
updateBook_ShouldUpdateBook()	4 ms
saveBook_ShouldSaveBook()	7 ms

Рисунок 17. Результаты тестов сервиса Book и его методов

9. Создание веб приложения используя Spring

(Мой выбор пал на веб чат)

- Выбор технологий для написания приложения

Spring MVC, Spring WebSoket для бэкенда и SockJS, JS, JQuery, ThymeLeaf для фронтенда.

- Конфигурация Spring проекта:

```

spring.datasource.url=jdbc:postgresql://localhost:5433/chatdb
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.username=postgres
spring.datasource.password=1212

spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.properties.hibernate.format_sql=true

```

- Определение и создание Entity базы данных в Spring

(Для работы с таблицами БД заодно используем валидацию данных, где это необходимо)

- Message Entity

(Представляет собой сообщения пользователей, отправленных ими кому-либо)

```

@Entity
@Setter
@Getter
@ToString
public class MessageEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "room_id")
    private RoomEntity room;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private UserEntity user;

    @Column(columnDefinition = "text")
    @NotBlank
    private String text;

    private Timestamp timestamp;
}

```

- Room Entity

(Представляет собой приватную комнату, в которой происходит общение между двумя пользователями)

```

@Entity
@Getter
@Setter
@ToString
@Table(name = "rooms")
public class RoomEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user1_id")
    private UserEntity user1;

    @ManyToOne
    @JoinColumn(name = "user2_id")
    private UserEntity user2;
}

```

- UserEntity

(Представляет собой таблицу, в которой будут храниться данные пользователей (id, пароль, логин, почта))

```

@Entity
@Setter
@Getter
@ToString
@Table(name = "users")
public class UserEntity {
    @ToString.Exclude
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

        private Long id;

        @NotBlank(message = "Username is mandatory")
        @Length(min = 5, message = "Nickname should have more than 5
characters")
        @Column(nullable = false, unique = true)
        private String username;

        @ToString.Exclude
        @Length(min = 6, message = "Password length must be more than
5 characters")
        @NotBlank(message = "Password is mandatory")
        @Column(nullable = false)
        private String password;

        @ToString.Exclude
        @Column(nullable = false, unique = true)
        @Email(message = "Please enter a correct email")
        private String email;
    }

```

- Конфигурация WebSocket для передачи сообщений с его помощью

```

@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements
WebSocketMessageBrokerConfigurer {
    @Override
    public void configureMessageBroker(MessageBrokerRegistry registry)
    {
        registry.setApplicationDestinationPrefixes("/app").enableSimpleBroker("/topic");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry)
    {
        registry.addEndpoint("/chat").withSockJS();
    }
}

```

- Конфигурация SpringSecurity для реализации авторизации и регистрации

```

@Configuration
@EnableWebSecurity
@AllArgsConstructor
public class SecurityConfig {

    private final UserDAO userDAO;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
throws Exception {
        http
            .csrf(AbstractHttpConfigurer::disable)
            .authorizeHttpRequests((authorize) ->
                authorize
                    .requestMatchers("/registration",
"/login*", "/webjars/**", "/css/**", "/js/**").permitAll()
                    .requestMatchers(HttpMethod.DELETE,
"/deletemessage**").authenticated()
                    .anyRequest().authenticated()
            )
    }
}

```



```

        .headers((headers) ->
            headers

.frameOptions(HeadersConfigurer.FrameOptionsConfig::sameOrigin)
        )
        .formLogin((formLogin) ->
            formLogin
                .loginPage("/login")
                .defaultSuccessUrl("/chat")

.failureUrl("/login?error=true").permitAll()
        )
        .logout((logout) ->
            logout
                .logoutRequestMatcher(new
AntPathRequestMatcher("/logout"))
                .logoutSuccessUrl("/login").permitAll()
        )
    );
    return http.build();
}

@Bean
public SecurityFilterChain securityFilterChainWebSock(HttpSecurity
http) throws Exception {
    http
        .csrf((csrf) -> {
            try {
                csrf

.disable().authorizeHttpRequests((authorize) ->
                authorize

.requestMatchers("/ws/**").permitAll()

.anyRequest().authenticated());
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        })
        .httpBasic(Customizer.withDefaults());
    return http.build();
}

@Bean
public BCryptPasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}

@Bean
public DaoAuthenticationProvider daoAuthenticationProvider() {
    final UserDetailsService userDetailsService = username -> {
        final UserEntity user = userDao.findByUsername(username);
        final GrantedAuthority userAuthority = new
SimpleGrantedAuthority("USER");
        return new User(user.getUsername(), user.getPassword(),
Collections.singletonList(userAuthority));
    };
    DaoAuthenticationProvider daoAuthenticationProvider = new
DaoAuthenticationProvider();
    daoAuthenticationProvider.setUserDetailsService(userDetailsService);

```

```
daoAuthenticationProvider.setPasswordEncoder(passwordEncoder());
    return daoAuthenticationProvider;
}
}
```

- Создание DAO для работы с базой данных в Spring
 - MessageDAO

```
@Repository
public interface MessageEntDAO extends
JpaRepository<MessageEntity, Integer> {
    @Query("SELECT m FROM MessageEntity m WHERE m.room.id =
:roomId " +
        "AND (m.user = :user OR m.room.user1 = :user OR
m.room.user2 = :user)")
    List<MessageEntity>
    findMessagesByRoomAndUser(@Param("roomId") Long roomId,
@Param("user") UserEntity user);

    @Query("SELECT m FROM MessageEntity m JOIN m.room r " +
        "WHERE ((r.user1.username = :currentUser AND
r.user2.username = :recipient) OR " +
        "(r.user1.username = :recipient AND r.user2.username
= :currentUser) " +
        "AND (m.user.username = :currentUser OR
m.user.username = :recipient) " +
        "ORDER BY m.timestamp ASC")
    List<MessageEntity>
    findMessagesByUsersNames(@Param("currentUser") String
currentUser, @Param("recipient") String recipient);

    @Query("SELECT m FROM MessageEntity m JOIN m.room r " +
        "WHERE ((r.user1.id = :currentUser AND r.user2.id =
:recipient) OR " +
        "(r.user1.id = :recipient AND r.user2.id =
:currentUser) " +
        "AND (m.user.id = :currentUser OR m.user.id =
:recipient) " +
        "ORDER BY m.timestamp ASC")
    List<MessageEntity>
    findMessagesByUsersId(@Param("currentUser") Long currentUser,
@Param("recipient") Long recipient);

    void deleteByTimestamp(Timestamp timestamp);

    MessageEntity findByTimestamp(Timestamp timestamp);
}
```

- RoomDAO

```
@Repository
public interface RoomDAO extends JpaRepository<RoomEntity,
Integer> {
    RoomEntity findByUser1_IdAndUser2_Id(Long user1Id, Long
user2Id);

    Boolean existsByUser1_IdAndUser2_Id(Long user1Id, Long
user2Id);

    @Query("SELECT c FROM RoomEntity c WHERE (c.user1.username =
:user1 AND c.user2.username = :user2) OR (c.user2.username =
:user1 AND c.user1.username = :user2)")
    RoomEntity
    findRoomEntityByUserNames(Long user1Id, Long user2Id);
}
```

```

findByUser1UsernameAndUser2UsernameOrUser2UsernameAndUser1Username(
    @Param("user1") String user1, @Param("user2") String user2);

    @Query("SELECT c FROM RoomEntity c WHERE (c.user1.id = :user1 AND c.user2.id = :user2) OR (c.user2.id = :user1 AND c.user1.id = :user2)")
    RoomEntity
    findByUserIdAndUser2IdOrUser2IdAndUserId(@Param("user1") Long user1, @Param("user2") Long user2);

    @Query("SELECT r FROM RoomEntity r WHERE r.user1.username = :username OR r.user2.username = :username")
    List<RoomEntity> findRoomsByUser(@Param("username") String username);
}

```

- UserDao

```

@Repository
public interface UserDao extends JpaRepository<UserEntity, Long> {
    UserEntity findByUsername(String username);

    boolean existsByUsername(String username);

    List<UserEntity> findAllByUsernameNot(String username);

    @Query("SELECT CASE " +
        "WHEN r.user1.username = :username THEN r.user2.id "
    +
        "WHEN r.user2.username = :username THEN r.user1.id "
    +
        "END " +
        "FROM RoomEntity r " +
        "WHERE r.user1.username = :username OR r.user2.username = :username")
    List<Long>
    findConversationParticipantIdsByUsername(@Param("username") String username);

    @Query("SELECT u FROM UserEntity u WHERE u.id <> :userId " +
        "AND u.id NOT IN (SELECT r.user1.id FROM RoomEntity r WHERE r.user2 = :user) " +
        "AND u.id NOT IN (SELECT r.user2.id FROM RoomEntity r WHERE r.user1 = :user) " +
        "AND u.username ILIKE :searchTerm%")
    List<UserEntity> findUsersWithoutRoomByUser(@Param("user") UserEntity user, @Param("userId") Long userId, @Param("searchTerm") String searchTerm);
}

```

- Распределение всей логики на эндпоинты (Mappings)

(для доступа к данным в браузере)

- Authentication controller для регистрации и логина

```

@Controller
@AllArgsConstructor
public class AuthController {

    private final UserService userService;
}

```

```

        @GetMapping("/registration")
        public String registerForm(Model model, UserEntity
userEntity) {
            model.addAttribute("user", userEntity);
            return "registration";
        }

        @PostMapping("/registration")
        public String registerSubmit(@Valid UserEntity userEntity,
BindingResult result, Model model) {
            if (result.hasErrors()) {
                return "registration";
            }
            userService.addUser(userEntity);
            return "redirect:/login";
        }

        @GetMapping("/login")
        public String loginForm() {
            return "login";
        }

        @GetMapping("chat/logout")
        public String logout() {
            return "redirect:/logout";
        }
    }

```

- o Message Controller для работы с сообщениями пользователей

```

@RestController
@AllArgsConstructor
public class MessageEntController {
    private final SimpMessagingTemplate simpMessagingTemplate;

    private final UserService userService;

    private final RoomService roomService;

    private final MessageEntService messageService;

    @MessageMapping("/chat/{recipient}")
    public void sendMessage(@DestinationVariable Long recipient,
MessageDto messageDto) {
        System.out.println("handling send message: " + messageDto
+ " to: " + recipient);

        MessageEntity message = new MessageEntity();

message.setRoom(roomService.findByUsersIds(messageDto.getSenderId()
(), messageDto.getRecipientId()));

message.setUser(userService.findById(messageDto.getSenderId()));
        message.setText(messageDto.getText());
        message.setTimestamp(messageDto.getTimestamp());

        if (userService.ifExistById(recipient)) {
            message = messageService.save(message);
            simpMessagingTemplate.convertAndSend(
                "/topic/messages/" + recipient,
MessagesDataDTO.getMessageDataDtoFromMessageEntity(message)
            );
        }
    }
}

```

```

    }
}

@PutMapping("/updatemessage")
public void updateMessage(@RequestParam Long timestamp,
@RequestBody UpdateMessageDTO messageDTO, Principal principal) {
    System.out.println("handling update message: " +
messageDTO + " to: " + timestamp );

    UserEntity user =
userService.findByUsername(principal.getName());

    MessageEntity message =
messageService.findByTimestamp(timestamp);

    if (message.getUser().equals(user)) {
        message.setText(messageDTO.getText());

        messageService.save(message);

        simpMessagingTemplate.convertAndSend(
            "/topic/updatemessage/" +
messageDTO.getRecipient(),
MessagesDataDTO.getMessageDataDtoFromMessageEntity(message)
        );
    }
}

@DeleteMapping("/deletemessage")
public void deleteMessage(@RequestBody DeleteMessageDTO
messageDTO) {
    System.out.println("delete");

    MessageEntity message =
messageService.findByTimestamp(messageDTO.getTimestamp());

    if (message != null) {
        messageService.delete(message);

        simpMessagingTemplate.convertAndSend(
            "/topic/deletemsg/" +
messageDTO.getRecipient(),
MessagesDataDTO.getMessageDataDtoFromMessageEntity(message)
        );
        simpMessagingTemplate.convertAndSend(
            "/topic/deletemsg/" +
messageDTO.getPrincipal(),
MessagesDataDTO.getMessageDataDtoFromMessageEntity(message)
        );
    }
}

@GetMapping("/getmessages")
public List<MessagesDataDTO> getPrivateMessages(@RequestParam
Long sender, @RequestParam Long recipient) {
    boolean isExists = userService.existsById(sender) &&
userService.existsById(recipient);

    if (isExists) {

```

```

System.out.println(messageService.findMessagesByUsersId(sender,
recipient).stream()
                .map(e -> e.getTimestamp().getTime())
                .toList());
        return messageService.findMessagesByUsersId(sender,
recipient).stream()

        .map(MessagesDataDTO::getMessageDataDtoFromMessageEntity)
                .toList();
    }
    return List.of(new MessagesDataDTO());
}
}

```

- Rooms Controller для работы с комнатами при помощи нужных запросов

```

@RestController
@AllArgsConstructor
public class RoomsController {
    private final RoomService roomService;

    private final SimpMessagingTemplate simpMessagingTemplate;

    @GetMapping("/fetchallrooms")
    public List<RoomDTO> fetchAll(Model model, Principal
principal) {
        return roomService.findRoomsByUser(principal.getName());
    }

    @GetMapping("/writetofound")
    public RoomDTO writeToFoundUser(Model model, @RequestParam
Long principalId, @RequestParam Long recipientId) {

simpMessagingTemplate.convertAndSend("/topic/newdialog/"+recipien
tId,principalId);
        return
RoomDTO.getRoomDtoFromRoom(roomService.insert(principalId,
recipientId));
    }
}

```

- Users Controller для получения информации о пользователях

```

@RestController
@CrossOrigin
@AllArgsConstructor
public class UsersController {

    private final UserService userService;

    @GetMapping("/fetchallusers")
    public List<UserDTO> fetchAll(Model model, Principal
principal, @RequestParam String searchTerm) {
        return
userService.findUsersWithoutRoomByUser(principal.getName(),
searchTerm);
    }

    @GetMapping("/fetchknownusers")
    public List<UserDTO> fetchKnown(Model model, Principal
principal) {
        return
userService.findAllUsersThatPrincipalKnows(principal.getName());
    }
}

```

```

    @GetMapping("/getprincipal")
    public UserDTO getPrincipal(Model model, Principal principal)
    {
        UserEntity user =
userService.findByUsername(principal.getName());

        return UserDTO.getUserDtoFromUser(user);
    }

    @GetMapping("/fetchuser")
    public UserDTO getUser(Model model, @RequestParam Long id) {
        return
UserDTO.getUserDtoFromUser(userService.findById(id));
    }
}

```

- Создание UI логин формы

(используя *Bootstrap* и *Thymeleaf* для доступа к бэкенду)

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <meta charset="utf-8">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <title>Login</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivrivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.m
in.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h1 class="text-center mb-4">Login</h1>
            <form th:action="@{/login}" method="post">
                <div class="form-group">
                    <label for="username">Имя пользователя:</label>
                    <input type="text" id="username" name="username"
required class="form-control">
                </div>
                <div class="form-group">
                    <label for="password">Пароль:</label>
                    <input type="password" id="password"
name="password" required class="form-control">
                </div>
                <button type="submit" class="btn btn-primary btn-
block">Войти</button>
            </form>
            <div class="text-center mt-4">
                <a href="/registration" class="btn btn-
link">Зарегистрироваться</a>
            </div>
        </div>
    </div>
</div>
</body>
</html>

```

- Создание формы регистрации по принципу логин формы

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <meta charset="utf-8">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <title>Registration</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.m
in.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h1 class="text-center mb-4">Registration</h1>
            <form th:action="@{/registration}"
th:object="${userEntity}" method="post">
                <div class="form-group">
                    <label for="username">Имя пользователя:</label>
                    <input type="text" id="username" name="username"
th:field="${username}" required class="form-control">
                    <span th:if="${#fields.hasErrors('username')}"
th:errors="${username}" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label for="email">Email пользователя:</label>
                    <input type="text" id="email" name="email"
th:field="${email}" required class="form-control">
                    <span th:if="${#fields.hasErrors('email')}"
th:errors="${email}" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label for="password">Пароль:</label>
                    <input type="password" id="password"
name="password" th:field="${password}" required class="form-control">
                    <span th:if="${#fields.hasErrors('password')}"
th:errors="${password}" class="text-danger"></span>
                </div>
                <button type="submit" class="btn btn-primary btn-
block">Зарегистрироваться</button>
            </form>
            <div class="text-center mt-4">
                <a href="/login" class="btn btn-link">Уже
зарегистрированы? Войти</a>
            </div>
        </div>
    </div>
</div>
</body>
</html>

```

- Создание UI и логики самого чата
 - chat.html

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Custom messenger</title>
    <script

```



```

src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.m
in.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.0/h
andlebars.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/list.js/1.1.1/list.mi
n.js"></script>
<!--      libs for stomp and sockjs-->
<script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-
client/1.4.0/sockjs.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/stomp.
min.js"></script>
<!--      end libs for stomp and sockjs-->

<link th:href="@{/styles/style.css}" rel="stylesheet">
<link rel="stylesheet" href="../../static/styles/style.css">
<script src="https://kit.fontawesome.com/2bce5db7d8.js"
crossorigin="anonymous"></script>
</head>
<body>
<div class="header">
<div class="search-wrapper">
<input type="text" id="userSearchInput" class="search-
input" placeholder="Search user...">
<div id="search-list-div" class="search-list">
<ul id="search-list">
</ul>
</div>
</div>
<div class="user-profile" id="object">

<span class="username" id="userName">n1kry</span>
<div id="menu" class="menu">
<ul>
<li class="log-out"><i class="fa-solid fa-door-
open" style="width: 10px; margin-right: 10px;"></i>Log Out</li>
</ul>
</div>
</div>
</div>
<div class="container clearfix">
<div class="row">
<div class="col-md-4">
<div class="people-list">
<div class="search">
<button class="btn btn-info" id="refreshBtn"
onclick="fetchKnown()"><i
class="fa-solid fa-arrow-rotate-
right"></i></button>
</div>
<ul class="list-group" id="usersList">
<a href="#" onclick="selectUser('n1kry')">
<li class="list-group-item list-group-
item-action mb-2 clearfix" data-toggle="list">
<div class="about">
<div id="userNameAppender_n1kry"
class="name">n1kry</div>
<div class="status">
<i class="fa fa-circle

```

```

offline"></i>
                                </div>
                            </div>
                        </li>
                    </a>
                    <a href="#" onclick="selectUser('crme029')">
                        <li class="list-group-item list-group-
item-action mb-2 clearfix" data-toggle="list">
                            <div class="about">
                                <div
id="userNameAppender_crme029" class="name">crme029</div>
                                <div class="status">
                                    <i class="fa fa-circle
offline"></i>
                                </div>
                            </div>
                        </li>
                    </a>
                </ul>
            </div>
        </div>
        <div class="col-md-8">
            <div class="chat">
                <div class="chat-header clearfix">
                    <link rel="icon" height="55px"
                        href="img/default-user-image.png"
                        width="55px"/>

                    <div class="chat-about">
                        <div class="chat-with"
id="selectedUserId"></div>
                        <div class="chat-num-messages"></div>
                    </div>
                </div> <!-- end chat-header -->
                <div class="chat-history" id="chat">
                    <ul class="list-unstyled" id="chat-history">
                        <li>
                            <div class="message-data">
                                <span class="message-data-
name"><i class="fa fa-circle online"></i> nlkry2</span>
                                <span class="message-data-
time">2:55 PM, Today</span>
                            </div>
                            <div class="message my-message">
                                hifg fdgfd g gfdg g
                                авыпавыпыпыпыпыывпыпываываывыпав
                            </div>
                        </li>
                        <li class="clearfix">
                            <div class="message-data align-
right">
                                <span class="message-data-
time">2:55 PM, Today</span> &nbsp; &nbsp;
                                <span class="message-data-
name">You</span> <i class="fa fa-circle me"></i>
                            </div>
                            <div class="message other-message
float-right">
                                hi
                            </div>
                        </li>
                    </ul>
                </div> <!-- end chat-history -->
            </div>
        </div>
    </div>

```

```

        <div class="context-menu">
            <ul>
                <li id="copy-button"><i class="fa-solid
fa-clone"></i>Copy</li>
                <li id="edite-button"><i class="fa-solid
fa-pen"></i>Edite</li>
                <li id="delete-button"><i class="fa-solid
fa-trash"></i>Delete</li>
            </ul>
        </div>
        <div class="chat-message clearfix">
            <textarea class="form-control" id="message-
to-send" name="message-to-send"
                placeholder="Type your message"
                rows="3"></textarea>
            <button id="sendBtn" class="btn btn-
primary">Send</button>
        </div> <!-- end chat-message -->
    </div> <!-- end chat -->
</div>
</div> <!-- end container -->

<script id="message-template" type="text/x-handlebars-template">
    <li class="clearfix">
        <div class="message-data align-right">
            <span class="message-data-time">{{time}},
Today</span> &nbsp; &nbsp;
            <span class="message-data-name">You</span> <i
class="fa fa-circle me"></i>
        </div>
        <div class="message other-message float-right"
id="{{myidentifier}}">
            {{messageOutput}}
        </div>
    </li>
</script>

<script id="message-response-template" type="text/x-handlebars-
template">
    <li>
        <div class="message-data">
            <span class="message-data-name"><i class="fa fa-
circle online"></i> {{userName}}</span>
            <span class="message-data-time">{{time}},
Today</span>
        </div>
        <div class="message my-message" id="{{otheridentifier}}">
            {{response}}
        </div>
    </li>
</script>

<script type="text/javascript"
th:src="@{/js/custom.js}"></script>
<script type="text/javascript" th:src="@{/js/chat.js}"></script>
<script>
    $(document).ready(function () {
        $("#object").mouseenter(function () {
            var menu = $("#menu");
            menu.show();
            var objectOffset = $(this).offset();
            menu.css({

```

```

        top: objectOffset.top + $(this).outerHeight() +
"px",
        left: objectOffset.left + "px"
    });
    }).mouseleave(function () {
        $("#menu").hide();
    });

});

let list;
$(document).ready(function () {
    $(".log-out").click(function () {
        window.location.href = "chat/logout";

    });

    $(document).on('click', function (e) {
        if (!$ (e.target).closest('#userSearchInput, #search-
list-div').length) {
            $('#search-list-div').hide();
        }
    });

    $('#search-list').on('click', 'li', function () {
        $('#search-list-div').hide();
    });
    $('#userSearchInput').on('input', function () {

        const searchTerm = $(this).val(); // Получение текста
из поля поиска

        if (searchTerm.length > 2) {
            $.get(url + "/fetchallusers?searchTerm=" +
searchTerm, function (response) {
                list = response;
                console.log('List', list)
            }).done(function () {
                $('#search-list-div').show();

                let usersListTemplate = '';
                console.log('Filtered', list)

                for (let i = 0; i < list.length; i++) {
                    usersListTemplate = usersListTemplate +
'<a href="#" onclick="writeToUser(\' ' + list[i].id + '\')">li
data-toggle="list">' + list[i].username + '</li></a>';
                }

                $('#search-list').html(usersListTemplate);
            });
        } else {
            $('#search-list-div').hide();
        }
    });
});
</script>
<script>
    $(document).ready(function () {
        var contextMenu; // Переменная для хранения текущего
открытого контекстного меню
        var messageContainer; // Переменная для хранения
родительского контейнера сообщения

```

```

        // Прикрепляем обработчик события contextmenu к каждому
сообщению
        $('message').on('contextmenu', function (e) {
            // Предотвращаем появление стандартного контекстного
меню
            e.preventDefault();

            // Закрываем предыдущее контекстное меню, если оно
было открыто
            if (contextMenu) {
                contextMenu.hide();
            }

            // Создаем новое контекстное меню
            contextMenu = $('#context-menu');
            const deleteButton = contextMenu.find('#delete-
button'); // Ищем кнопку "Удалить" внутри контекстного меню
            const copyButton = contextMenu.find('#copy-button');
            const editButton = contextMenu.find('#edit-
button');

            // Получаем родительский контейнер сообщения
            messageContainer = $(this).closest('li');
            console.log(messageContainer);

            // Прикрепляем обработчик события click к кнопке
"Удалить"
            deleteButton.on('click', function () {
                // Удаляем текущий элемент <li> из chat-history
                messageContainer.remove();
                contextMenu.hide();
            });
            copyButton.on('click', function () {
                navigator.clipboard.writeText(messageContainer.context.innerText)
                .catch(function(error) {
                    console.error('Ошибка при копировании текста:
', error);
                });
            });

            editButton.on('click', function () {
                const textArea = $('#message-to-send');
                const saveBtn = $('#sendBtn');

                textArea.text(messageContainer.context.innerText);
                saveBtn.text('save');
            });

            // Позиционируем контекстное меню относительно щелчка
мышь
            var posX = e.pageX;
            var posY = e.pageY;
            contextMenu.css({top: posY, left: posX});

            // Показываем контекстное меню
            contextMenu.show();

            // Закрываем контекстное меню при щелчке вне его
области
            $(document).on('click', function () {
                contextMenu.hide();
            });

```

```

    });
  });
</script>
</body>
</html>

```

○ custom.js

```

let $chatHistory; // Переменная для хранения элемента истории чата
let $button; // Переменная для хранения кнопки отправки сообщения
let $textarea; // Переменная для хранения текстового поля ввода сообщения
let $chatHistoryList; // Переменная для хранения списка сообщений чата

function init() {
  cacheDOM(); // Вызов функции для кэширования элементов DOM
  bindEvents(); // Вызов функции для привязки событий
}

function bindEvents() {
  $button.on('click', addMessage.bind(this)); // Привязка события клика на кнопке отправки сообщения
  $textarea.on('keyup', addMessageEnter.bind(this)); // Привязка события нажатия клавиши Enter в текстовом поле ввода сообщения
  $deleteButton.on('click', deleteMessage.bind(this))
  $copyButton.on('click', copyMessage.bind(this))
  $editButton.on('click', editMessage.bind(this))
}

function cacheDOM() {
  $chatHistory = $('.chat-history'); // Кэширование элемента истории чата
  $button = $('#sendBtn'); // Кэширование кнопки отправки сообщения
  $textarea = $('#message-to-send'); // Кэширование текстового поля ввода сообщения
  $chatHistoryList = $chatHistory.find('ul'); // Кэширование списка сообщений чата
}

// Переменная для хранения текущего открытого контекстного меню
function copyMessage() {
  navigator.clipboard.writeText(messageContainer.context.innerText)
  .catch(function (error) {
    console.error('Ошибка при копировании текста: ', error);
  });
}

function deleteMessage() {
  // Delete the message here
  $.ajax({
    url: '/deletemessage',
    type: 'DELETE',
    data: JSON.stringify({
      timestamp: messageContainer.find('[id]').attr('id'),
      recipient: selectedUser.id,
      principal: principal.id
    }),
    contentType: 'application/json',

```

```

        success: function () {
            contextMenu.hide();
        },
        error: function (error) {
            console.error('Error:', error);
        }
    });
}

function editMessage() {
    console.log('edit')
    const textArea = $('#message-to-send');
    const saveBtn = $('#sendBtn');

    textArea.val(messageContainer.context.innerText)
    textArea.focus();
    saveBtn.text('save');
}

let contextMenu = $('.context-menu');
let messageContainer;
const $deleteButton = contextMenu.find('#delete-button');
const $editeButton = contextMenu.find('#edite-button');
const $copyButton = contextMenu.find('#copy-button');

function addContextMenu(id) {
    // Attach contextmenu event handler to each message
    $('#${id}`).on('contextmenu', function (e) {
        // Prevent the default context menu from appearing
        e.preventDefault();

        // Закрытие предыдущего контекстного меню, если есть
        if (contextMenu) {
            contextMenu.hide();
        }

        // Создание нового контекстного меню
        messageContainer = $(this).closest('li');

        // Позиционирование контекстного меню относительно
        // нажатого сообщения
        const posX = e.pageX;
        const posY = e.pageY;
        contextMenu.css({top: posY, left: posX});

        console.log($(this).find('div[class]'))
        console.log($(this).hasClass('my-message'))

        if ($(this).hasClass('my-message')) {
            $editeButton.hide();
        } else {
            $editeButton.show();
        }

        contextMenu.show();
    });
}

// Закрытие контекстного меню при клике вне него
$(document).ready(function () {
    $(document).on('click', function () {
        contextMenu.hide();
    });
});

```

```

    });
  });

function render(sender, recipient) {
    let templateResponse = Handlebars.compile($("#message-
response-template").html()); // Компиляция шаблона для
отображения полученных сообщений
    let template = Handlebars.compile($("#message-
template").html()); // Компиляция шаблона для отображения
отправленных сообщений

    console.log(sender, recipient)

    setTimeout(function () {
        $.get(url + "/getmessages?sender=" + sender.id +
"&recipient=" + recipient.id, function (response) {
            let messages = response; // Получение списка
сообщений
            console.log(messages);
            for (let i = 0; i < messages.length; i++) {
                if (messages[i].user.username ===
principal.username) {
                    $chatHistoryList.append(template({
                        messageOutput: messages[i].text,
                        time: getTime(messages[i].timestamp),
                        myidentifier:
Date.parse(messages[i].timestamp).valueOf()
                    })); // Отображение отправленных сообщений в
чате
                } else {
                    $chatHistoryList.append(templateResponse({
                        response: messages[i].text,
                        time: getTime(messages[i].timestamp),
                        userName: selectedUser.username,
                        otheridentifier:
Date.parse(messages[i].timestamp).valueOf()
                    })); // Отображение полученных сообщений в
чате
                }
            }
            scrollToBottom(); // Прокрутка до конца истории чата
        })
    }).bind(this, 200);
}

function sendMessage(message) {
    let currentTime = new Date();
    sendMsg(principal, message, currentTime); // Отправка
сообщения
    scrollToBottom(); // Прокрутка до конца истории чата
    if (message.trim() !== '') {
        let template = Handlebars.compile($("#message-
template").html());
        let context = {
            messageOutput: message,
            time: getTime(currentTime),
            myidentifier: currentTime.getTime()
        };
    }
}

```



```

        $chatHistoryList.append(template(context)); //
Отображение отправленного сообщения в чате
        addContextMenu(currentTime.valueOf());
        scrollToBottom(); // Прокрутка до конца истории чата
        $textarea.val(''); // Очистка текстового поля ввода
сообщения
    }
}

function updateMessage(val) {
    $button.text('send');
    const timestamp = messageContainer.find('[id]').attr('id');
    const text = val;

    updateMsg(text, timestamp);

    $('#'+ timestamp).text(text);
    $textarea.val('');
}

function liveRender(message, userName, timestamp) {
    scrollToBottom(); // Прокрутка до конца истории чата
    // responses
    let templateResponse = Handlebars.compile($("#message-
response-template").html());
    let contextResponse = {
        response: message,
        time: getTime(timestamp),
        userName: userName,
        otheridentifier: Date.parse(timestamp).valueOf()
    };

    setTimeout(function () {

        $chatHistoryList.append(templateResponse(contextResponse)); //
Отображение полученного сообщения в чате
        scrollToBottom(); // Прокрутка до конца истории чата
        addContextMenu(new Date(timestamp).valueOf());
    }.bind(this), 200);
}

function scrollToBottom() {
    $chatHistory.scrollTop($chatHistory[0].scrollHeight); //
Прокрутка до конца истории чата

    console.log($chatHistory.scrollTop($chatHistory[0].scrollHeight))
;
}

function getTime(timestamp) {
    console.log(timestamp)
    return new
Date(timestamp).toLocaleTimeString().replace(/([\d]+:[\d]{2})(:[\d]{2})(.*)/, "$1$3"); // Получение текущего времени
}

function addMessage() {
    console.log($button.text().toUpperCase())
    if ($button.text().toUpperCase() === 'send'.toUpperCase()) {
        sendMessage($textarea.val()); // Добавление сообщения
    } else {
        updateMessage($textarea.val(), messageContainer);
    }
}

```

```

    }
}

function addMessageEnter(event) {
    // enter was pressed
    if (event.keyCode === 13) {
        addMessage(); // Добавление сообщения при нажатии клавиши
    }
}

init(); // Инициализация приложения при загрузке страницы

```

o chat.js

```

const url = 'http://localhost:8080'; // URL-адрес сервера, с
которым будет установлено соединение

let stompClient; // Объект StompClient для обмена сообщениями по
протоколу STOMP
let selectedUser; // Имя выбранного пользователя для чата
let principal; // Переменная для хранения имени текущего
пользователя

registration(); // Вызов функции регистрации

let socket = new SockJS(url + '/chat'); // Создание нового
WebSocket-соединения

// Функция для подключения к чату
function connectToChat(principal) {
    console.log("connecting to chat...") // Вывод сообщения о
    попытке подключения к чату
    stompClient = Stomp.over(socket); // Создание объекта
    StompClient для управления соединением
    stompClient.connect({}, function (frame) {
        console.log("connected to: " + frame); // Вывод сообщения
    об успешном подключении к чату
    stompClient.subscribe("/topic/messages/" + principal.id,
    function (response) {
        let data = JSON.parse(response.body); // Разбор
        полученных данных в формате JSON
        console.log("Data", data);
        if (!selectedUser) {
            $('#userNameAppender_' +
            data.user.id).append('<span id="newMessage_' + data.user.id + '"
            style="color: red">+1</span>'); // Отображение индикатора нового
            сообщения для пользователя в списке пользователей
        } else {
            if (selectedUser.username === data.user.username)
            {
                liveRender(data.text, data.user.username,
                data.timestamp); // Вызов функции для отображения полученного
                сообщения в чате
            } else {
                $('#userNameAppender_' +
                data.user.id).append('<span id="newMessage_' + data.user.id + '"
                style="color: red">+1</span>'); // Отображение индикатора нового
                сообщения для пользователя в списке пользователей
            }
        }
    });
    stompClient.subscribe("/topic/newdialog/" + principal.id,

```

```

function (r) {
    console.log(r)
    $.get(url + "/fetchuser?id=" + r.body, function
(response) {
        users.push(response);
        appendUsers(response.id, response.username);
    });
})
stompClient.subscribe('/topic/updatesmessage/' +
principal.id, function (r) {
    console.log(r)
    const data = JSON.parse(r.body);
    console.log(selectedUser.id === Number(data.user.id))
    if (selectedUser.id === Number(data.user.id)) {
        $('#'+
Date.parse(data.timestamp).valueOf()).text(data.text);
    }
})
stompClient.subscribe('/topic/deletemsg/' + principal.id,
function (r) {
    const data = JSON.parse(r.body);

    if (!!$('#${Date.parse(data.timestamp)}')) {
        $('#${Date.parse(data.timestamp)}').parent().remove();
    }
})
});
}

// Функция для отправки сообщения
function sendMsg(from, text, timestamp) {
    const message = {
        senderId: from.id,
        recipientId: selectedUser.id,
        text: text,
        timestamp: timestamp
    }
    console.log('Message ', message);
    stompClient.send("/app/chat/" + selectedUser.id, {},
JSON.stringify(message));
}

function updateMsg(text, timestamp) {
    console.log('Message ', text);

    const message = {
        recipient: selectedUser.id,
        text: text,
    };

    fetch('/updatesmessage?timestamp=' + timestamp, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(message),
    })
    .then(response => {
        if (response.ok) {
            console.log('Message updated successfully');
        } else {
            throw new Error('Error updating message');
        }
    })
}

```

```

    }
  })
  .catch(error => {
    console.error('Error:', error);
  });
}

// Функция для регистрации пользователя
function registration() {
  $.get(url + "/getprincipal", function (response) {
    principal = response; // Получение имени текущего
    пользователя

    console.log('Principal -> ', principal)

    $('#userName').text(principal.username); // Отображение
    имени текущего пользователя

    connectToChat(principal); // Подключение к чату с
    использованием имени текущего пользователя

    fetchKnown(); // Получение списка пользователей
  });
}

// Функция для выбора пользователя для чата
function selectUser(userId) {
  console.log("selecting users: " + userId); // Вывод
  выбранного пользователя в консоль
  console.log()

  selectedUser = users.find(u => u.id === Number(userId)); //
  Установка выбранного пользователя
  console.log('Selected user', selectedUser)

  let isNew = document.getElementById("newMessage_" +
  selectedUser.id) !== null; // Проверка, есть ли у выбранного
  пользователя новые сообщения
  if (isNew) {
    let element = document.getElementById("newMessage_" +
    selectedUser.id);
    element.parentNode.removeChild(element); // Удаление
    индикатора нового сообщения для выбранного пользователя
  }
  $('#selectedUserId').html('');
  $('#selectedUserId').append('Chat with ' +
  selectedUser.username); // Отображение выбранного пользователя
  для чата
  $('#chat-history').html('').removeClass('unselected'); //
  Очистка истории чата
  $('#chat-message').show();

  render(principal, selectedUser); // Отображение сообщений
  между текущим пользователем и выбранным пользователем
}

let users; //массив объектов user (тех с которыми общается
principal)

// Функция для получения списка всех пользователей
function fetchKnown() {
  $.get(url + "/fetchknownusers", function (response) {
    users = response; // Получение списка пользователей
  });
}

```

```

        console.log('Fetch', users)
        $('#usersList').html(''); // Отображение списка
пользователей
        $('#selectedUserId').html('');
        $('#chat-history').html('Chose someone to start chatting
:').addClass('unselected');
        $('.chat-message').hide();

        selectedUser = null;

        for (let i = 0; i < users.length; i++) {
            appendUsers(users[i].id, users[i].username)
        }
    }).done(function () {
        $('#usersList').off('click', 'li').on('click', 'li',
function (e) {
            const current =
document.getElementsByClassName("selected");
            if (current.length > 0) {
                current[0].classList.remove("selected");
            }
            this.classList.add("selected"); // Выделение
выбранного пользователя в списке пользователей
        });
    });
}

function writeToUser(id) {

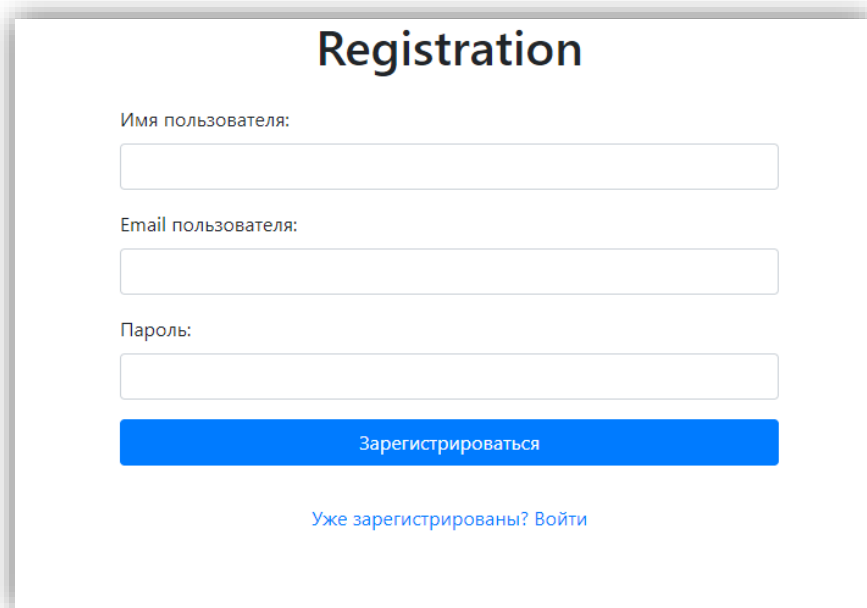
    $.get(url + '/writetofound?principalId=' + principal.id +
'&recipientId=' + id, function (response) {
        let room = response;
        console.log('Room', room)

    })
    console.log("write to user " + id)
    console.log("write to user1 " + list.find(u => u.id ===
Number(id)).id)
    let user = list.find(u => u.id === Number(id));
    appendUsers(user.id, user.username)
    users.push(list.find(u => u.id === Number(id)));
    console.log('Pushed users', users)
}

function appendUsers(id, username) {
    let usersTemplateHTML = '<a href="#" onclick="selectUser(\' '
+ id + '\')"><li class="list-group-item list-group-item-action
mb-2 clearfix" data-toggle="list">\n' +
        '
            <div class="about">\n' +
                '
                    <div id="userNameAppender_' + id +
'" class="name">' + username + '</div>\n' +
                '
                    <div class="status">\n' +
                    '
                        <i class="fa fa-circle
offline"></i>\n' +
                    '
                        </div>\n' +
                '
                    </div>\n' +
            '
                </li></a>';
    $('#usersList').append(usersTemplateHTML);
}

```

- Тестирование и демонстрация приложения:
 - Регистрация



Registration

Имя пользователя:

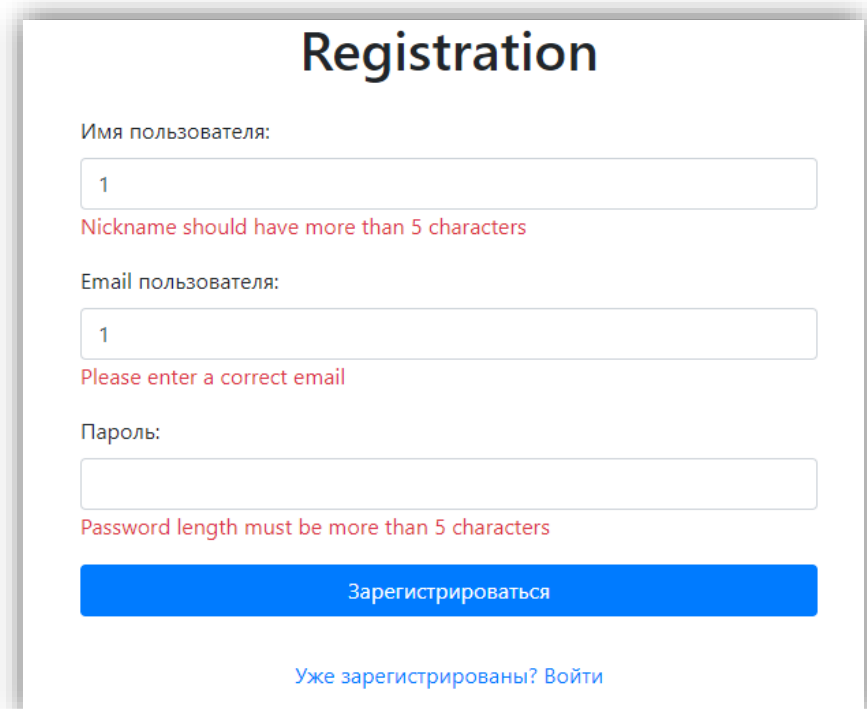
Email пользователя:

Пароль:

Зарегистрироваться

[Уже зарегистрированы? Войти](#)

Рисунок 18. Форма регистрации



Registration

Имя пользователя:

Nickname should have more than 5 characters

Email пользователя:

Please enter a correct email

Пароль:

Password length must be more than 5 characters

Зарегистрироваться

[Уже зарегистрированы? Войти](#)

Рисунок 19. Проверка валидации во время регистрации

Registration

Имя пользователя:

Email пользователя:

Пароль:

Password length must be more than 5 characters

Зарегистрироваться

[Уже зарегистрированы? Войти](#)

Рисунок 20. Проверка валидации для одного поля

*После регистрации нас перекидывает на логин форму**

- Вход в аккаунт

Login

Имя пользователя:

Пароль:

Войти

[Зарегистрироваться](#)

Рисунок 21. Форма логина с неправильным паролем

localhost:8080/login?error=true

Рисунок 22. Результат валидации неверного пароля

*После успешного входа нас перекинет на сам чат**

- Чат

- Первоначальная страница:

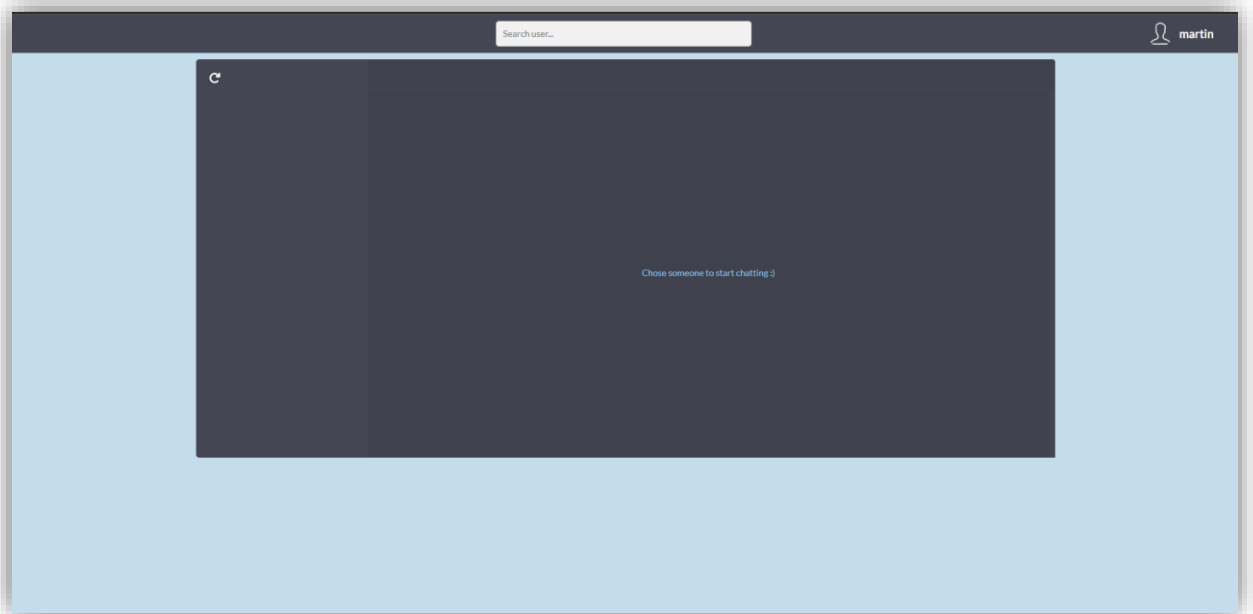


Рисунок 23. Главная страница чата

- Поиск собеседника

(Для этого в поисковой строке достаточно ввести первые 3 символа никнейма вашего друга и вам высветится выпадающий список всех пользователей, которые подходят по данному началу)

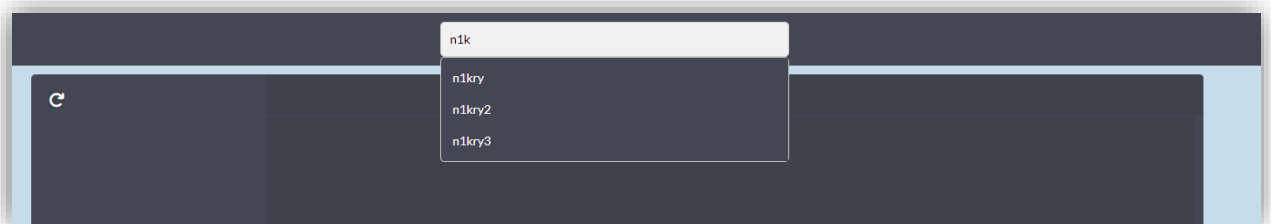


Рисунок 24. Поиск новых собеседников

- Добавление нужного пользователя в друзья

(Для этого нужно нажать на нужного человека (В данном случае я нажму на n1kry и у обоих пользователей появится его новый собеседник в левом списке с собеседниками))

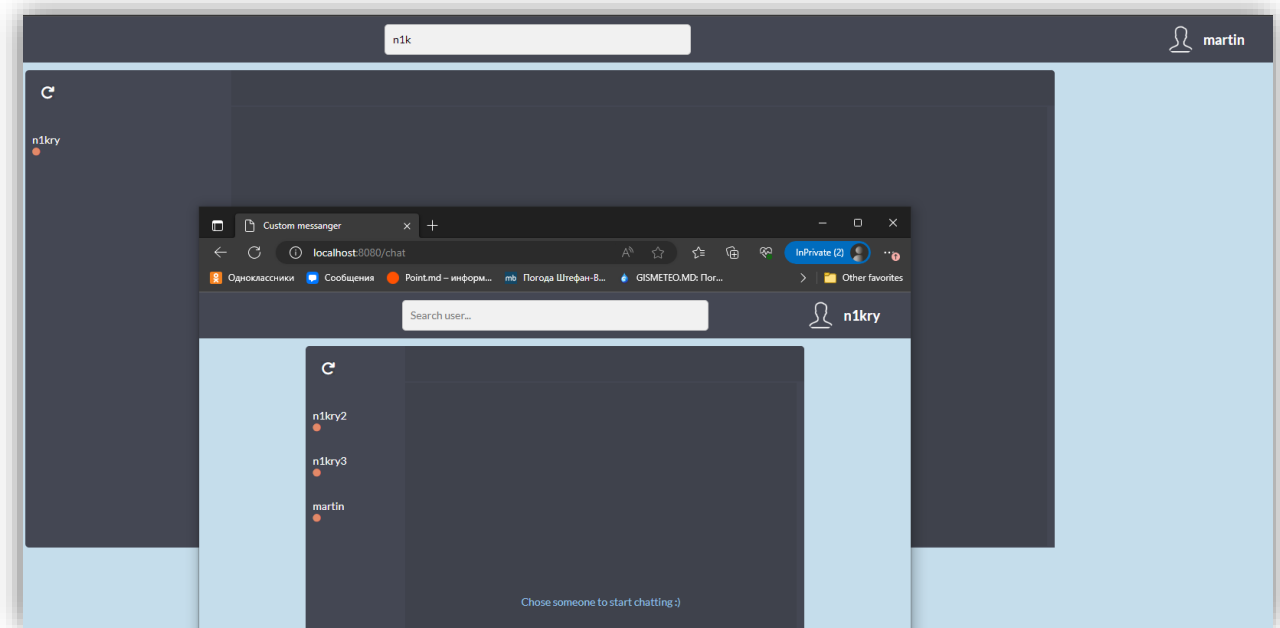


Рисунок 25. Отображение нового собеседника в меню пользователей слева

- Отправка сообщения

(Теперь мы можем смело писать новому другу и при новом сообщении на другой стороне будет отображение нового сообщения)

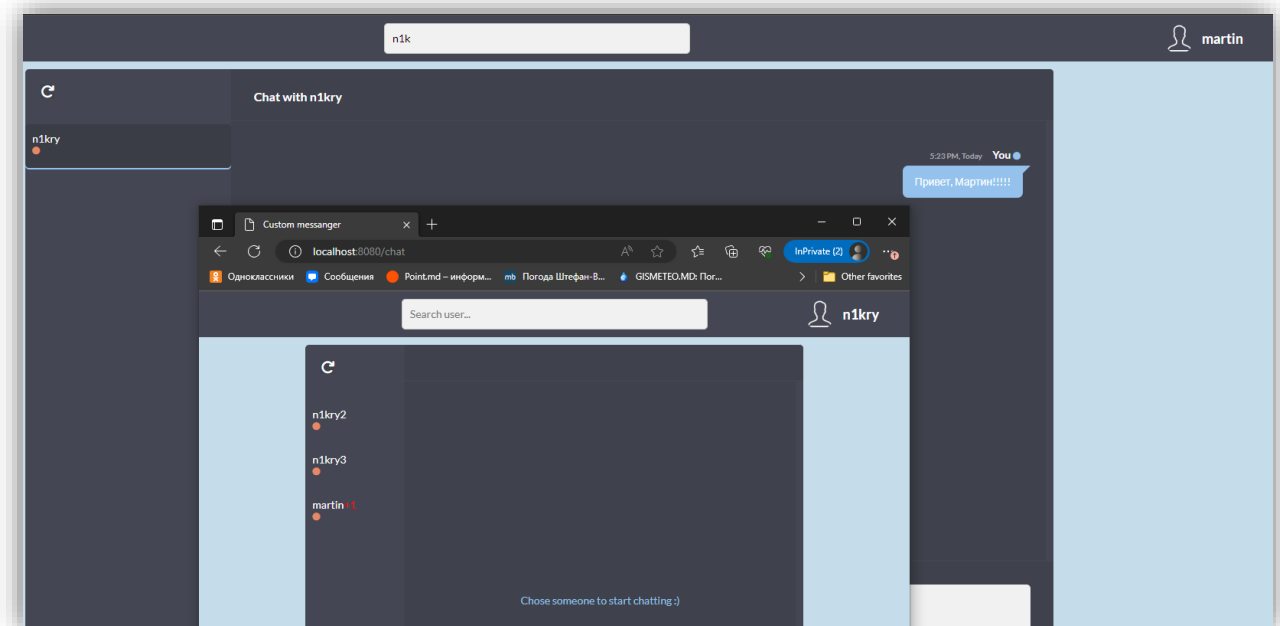


Рисунок 26. Отправка сообщения и уведомление у собеседника

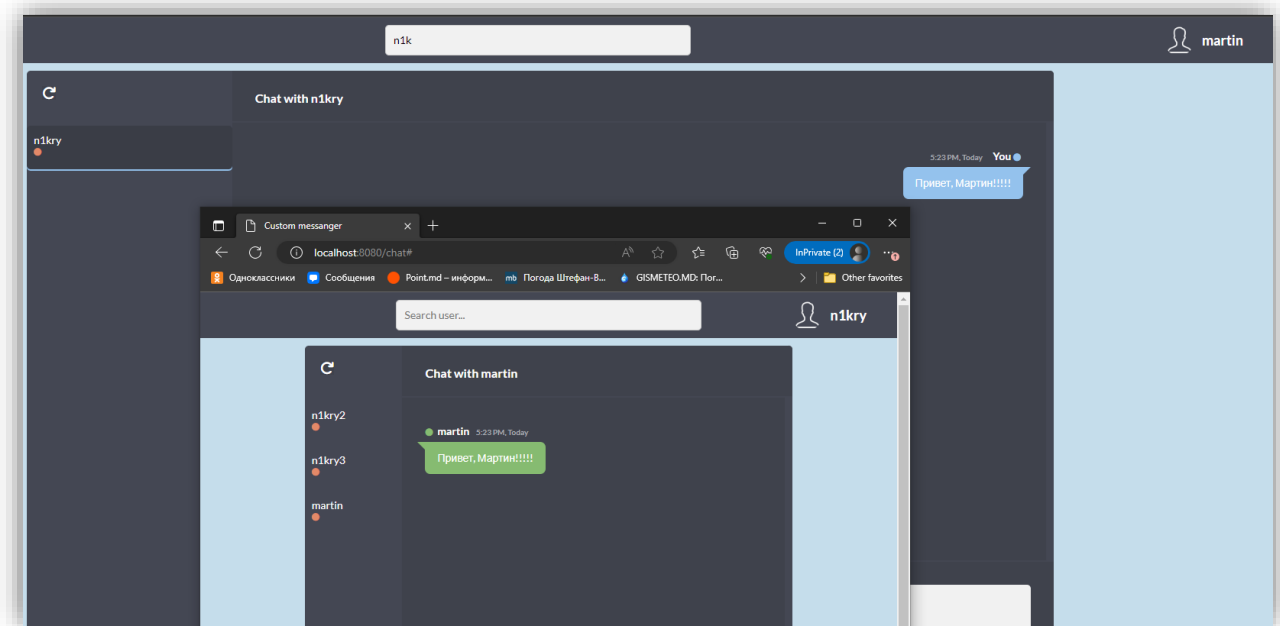


Рисунок 27. Просмотр полученного сообщения

- Ответ на сообщение

(Так же мы без проблем можем ответить ему, и он мгновенно получит наше сообщение и увидит его)

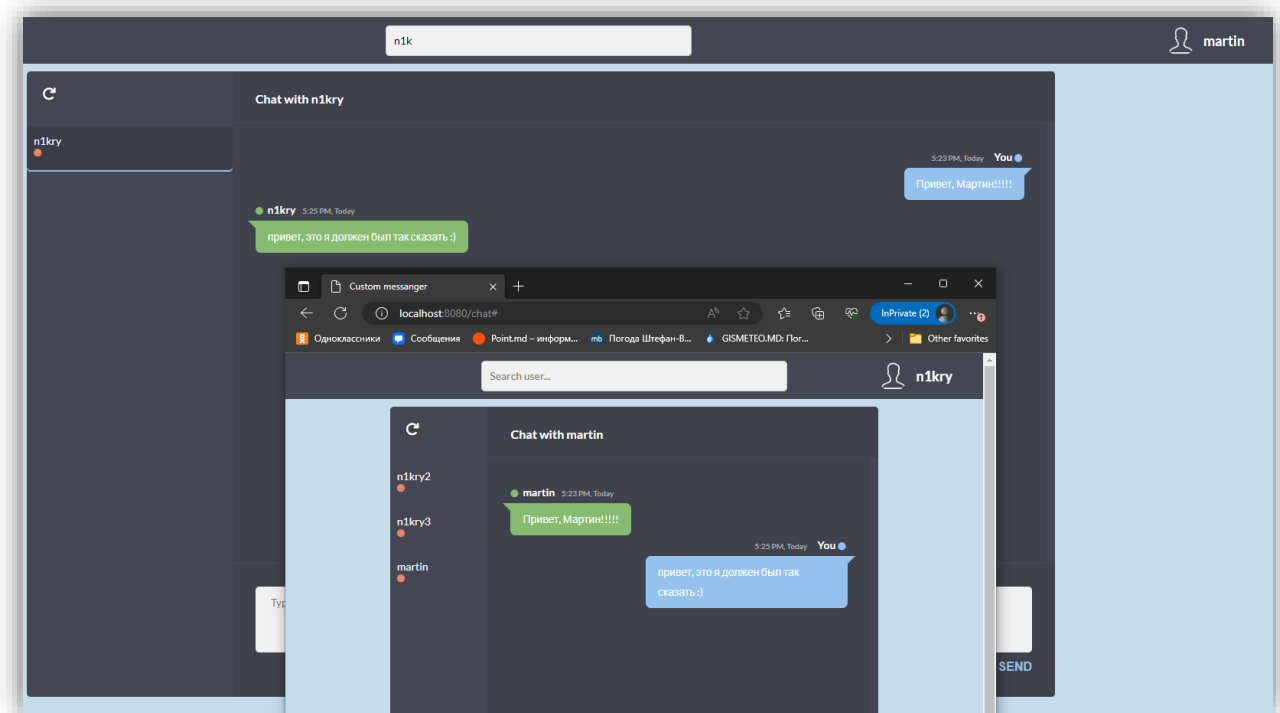


Рисунок 28. Ответ на полученное сообщение с автоматическим обновлением у собеседника

- Редактирование сообщения

(Еще мы можем редактировать наше сообщение если мы в нем ошиблись для этого нужно нажать правой кнопкой мыши на него и выбрать нужную опцию)

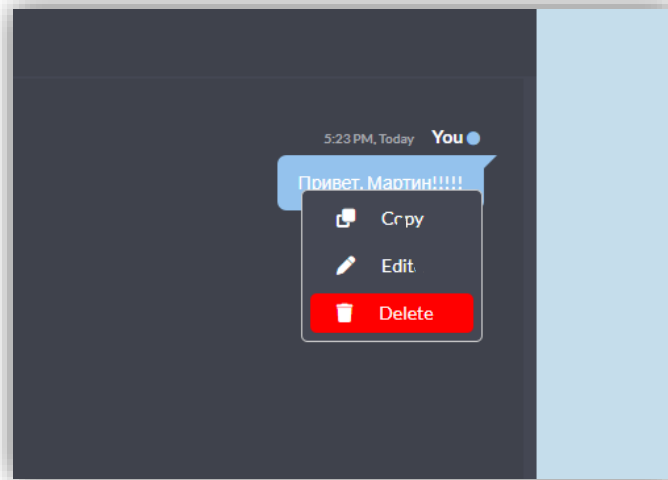


Рисунок 29. Контекстное меню при щелчке на свое сообщение

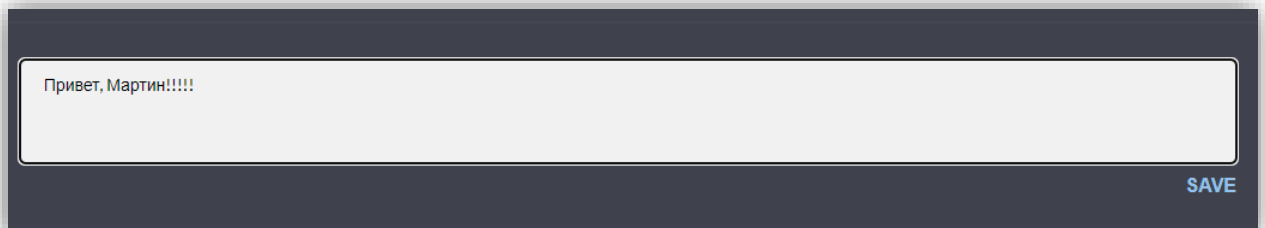


Рисунок 30. После нажатия на кнопку Edit

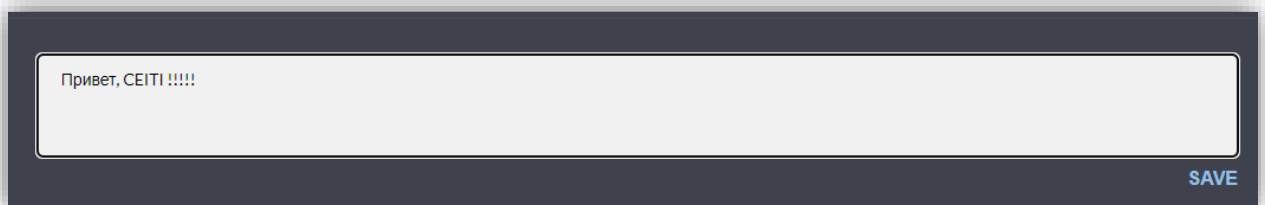


Рисунок 31. Изменение сообщения

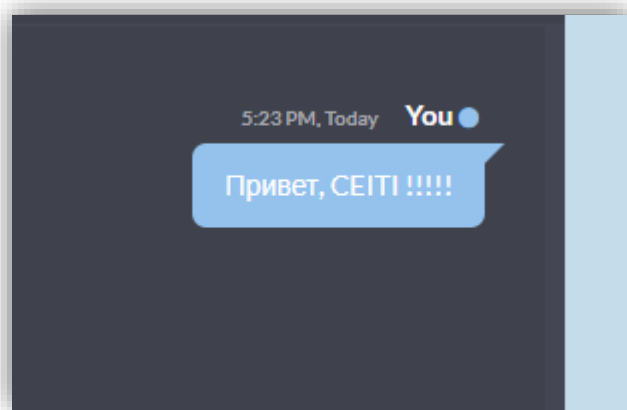


Рисунок 32. Измененное сообщение после нажатия кнопки SAVE или нажатия Enter на клавиатуре

- Удаление сообщения

(А если сообщение было отправлено по ошибке мы можем его запросто удалить, а также есть возможность удалить сообщение собеседника если оно вам не по душе для этого нужно все, то же контекстное меню)

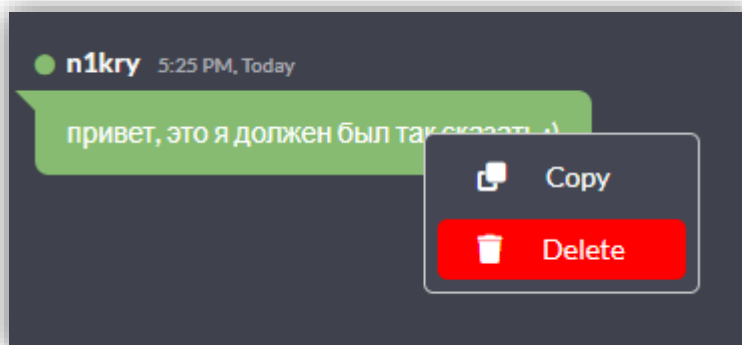
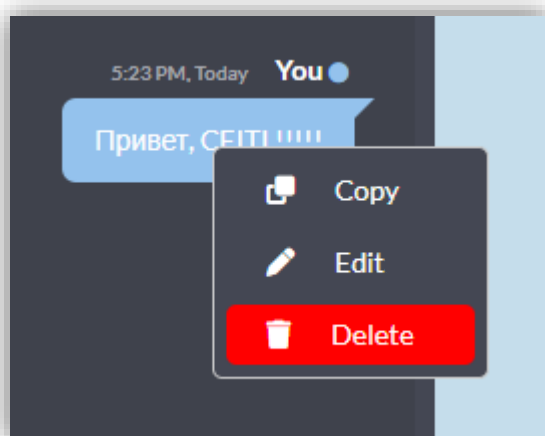


Рисунок 33. Контекстное меню сообщения собеседника



Удаляем сообщение собеседника и автоматически оно у него пропадет, как и у нас:

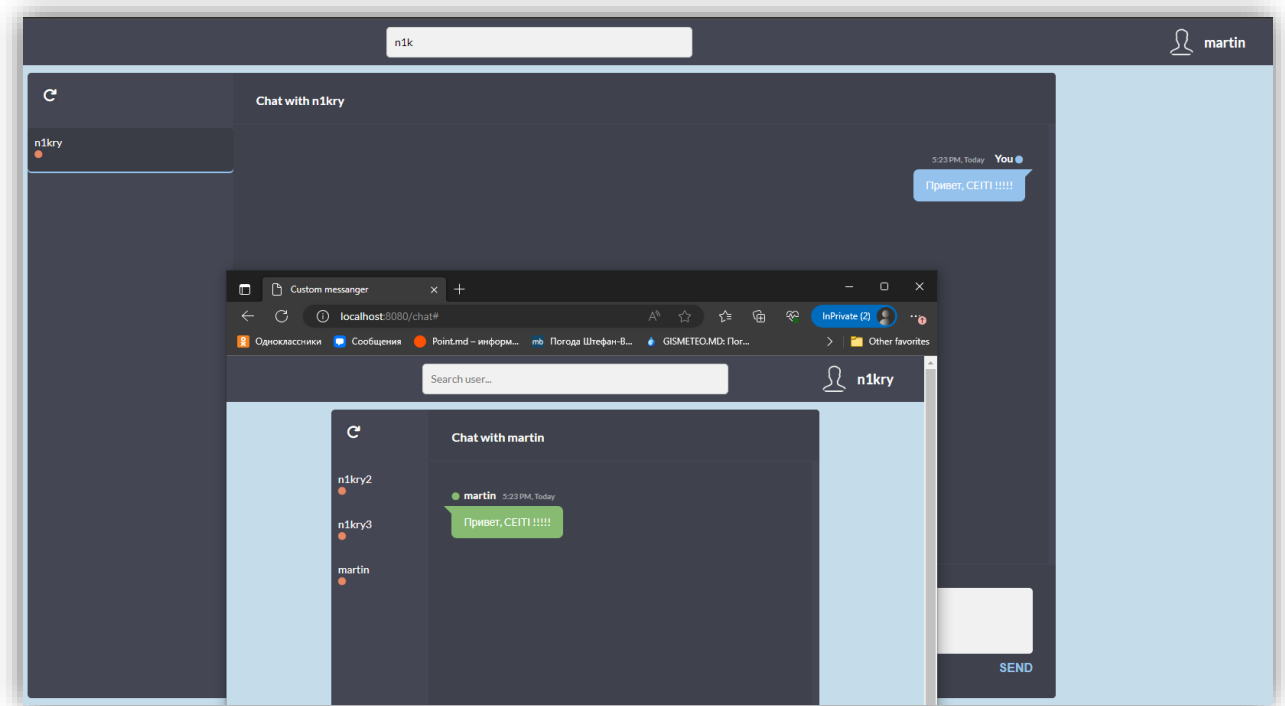


Рисунок 34. Состояние чата после удаления сообщения собеседника

■ Выход из аккаунта

(А когда мы хотим войти в другой аккаунт нам достаточно навести на своего пользователя и нажать кнопку *Log Out* в правой верхней части)

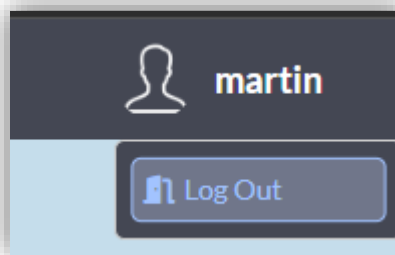


Рисунок 35. Кнопка выхода при наведении курсора на профиль

- В это время в БД

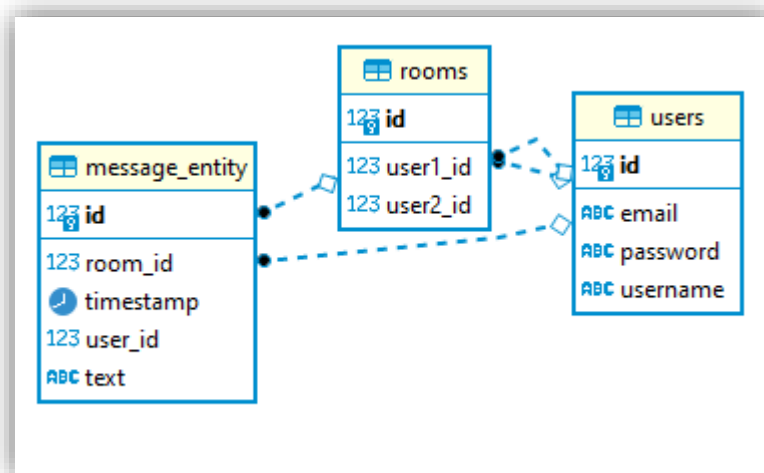


Рисунок 36. Схема БД

	123 id	123 room_id	timestamp	123 user_id	ABC text
1	1	1	2023-06-14 17:07:52.866	1	Hello1
2	2	1	2023-06-14 17:07:52.871	2	Hello2
3	3	3	2023-06-14 17:23:23.197	6	Привет, CEIT !!!!!

Рисунок 37. Таблица Сообщений

	123 id	123 user1_id	123 user2_id
1	1	1	2
2	2	1	3
3	3	6	1

Рисунок 38. Таблица комнат

	123 id	ABC email	ABC password	ABC username
1	1	1@mail.com	\$2a\$10\$9kqOs8LfPOKuudMHgEsv2e2.PzhJGu9iwkUdAB8MhEPdSvDjlK6Pm	n1kry
2	2	2@mail.com	\$2a\$10\$FhEoeJVetkcpTRfhQKsavu2Dvk3j3KqfZADQsR4erfedtRqVW3dGS	n1kry2
3	3	3@mail.com	\$2a\$10\$w3WH0UCwAhYEI6SJ5RREz.cJAP0gCb1/pfMpZ4cSTLHJUDizSf8OO	n1kry3
4	4	4@mail.com	\$2a\$10\$6.ILE/L9xUGf0/oLVkCWA.uw2UfnHQnnTR9t2mzdRTKD8mc6Xcfbq	friend1
5	5	5@mail.com	\$2a\$10\$NPK7rJ0e6ke78EVgEKMEsefGYR7JgtwwjYY3YXofvSlrDoQJK5GgK	friend2
6	6	iden@mail.com	\$2a\$10\$eSgcdVnsu1pnAVVYW4I5o.toWUX9.DII9ZOdTrQ/8XpL5gwflNog.	martin

Рисунок 39. Таблица пользователей

Общие замечания

В целом, приложение "Веб чата" разработано с использованием современных технологий и инструментов, имеет отзывчивый дизайн и полный функционал для удобного использования пользователями. Однако, возможны некоторые улучшения в области безопасности и производительности, которые могут быть рассмотрены в будущих обновлениях.

Выводы

Практика, предложенная Cedacri International, предоставила мне возможность выполнить разнообразные задачи, связанные с разработкой программного обеспечения и тестированием. В результате выполнения этих заданий я получил практический опыт в области структур данных, использования инструментов разработки, баз данных и фреймворков, таких как Maven, SQL, Jasper Reports, JUnit, Hibernate и Spring.

Выполнение этих задач помогло мне углубить свои знания и навыки в соответствующих областях, ознакомиться с передовыми технологиями и методиками разработки программного обеспечения, а также понять лучшие практики в написании чистого кода и использовании рефакторинга.

В процессе выполнения заданий я смог применить полученные знания на практике, развить навыки сотрудничества с командой и решать реальные проблемы, связанные с разработкой программного обеспечения.

В целом, выполнение этих задач позволило мне узнать много нового, расширить свои навыки и подготовиться к будущим вызовам в области программирования и разработки программного обеспечения. Эта практика оказалась очень ценной и полезной для моего профессионального развития

Библиография

- <https://docs.oracle.com/javase/8/docs/api/java/util/Stack.html>
- <https://maven.apache.org/what-is-maven.html>
- <https://www.mkymong.com/tutorials/maven-tutorials/>
- <https://howtodoinjava.com/maven/>
- <https://dzone.com/articles/maven-multi-module-project-with-versioning>
- <http://sqlzoo.net/>
- <http://community.jaspersoft.com/wiki/getting-started-ireport-designer>
- <http://community.jaspersoft.com/wiki/basic-report-components>

- https://dynamicreports.lbayer.com/examples/simplereport_step01/
- <https://www.mkyong.com/java/reporting-in-java-using-dynamicreports-and-jasperreports/>
- <http://www.vogella.com/tutorials/JUnit/article.html>
- <https://www.oracle.com/technetwork/java/codeconventions-150003.pdf>
- <https://google.github.io/styleguide/javaguide.html>
- <https://refactoring.guru/smells/smells>
- <http://www.garshol.priv.no/blog/105.html>
- <https://www.mkyong.com/tutorials/hibernate-tutorials/>
- <https://www.mkyong.com/tutorials/spring-tutorials/>
- <https://spring.io/guides/gs/messaging-stomp-websocket/>