

СОДЕРЖАНИЕ

Перечень условных обозначений, символов и терминов	6
Введение.....	8
1 Анализ и моделирование предметной области программного средства.....	9
1.1 Описание предметной области	9
1.2 Разработка функциональной модели предметной области	10
1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований.....	14
1.4 Разработка информационной модели предметной области	15
1.5 Модели представления программного средства и их описание	18
1.6 Диаграмма последовательности	18
1.7 Диаграмма деятельности	19
1.8 Диаграмма развертывания.....	21
2 Проектирование и конструирование программного средства	23
2.1 Постановка задачи	23
2.2 Архитектурные решения	23
2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства	27
2.4 Проектирование пользовательского интерфейса	29
2.5 Обоснование выбора компонентов и технологий для реализации программного средства	30
3 Тестирование и проверка работоспособности программного средства.....	32
4 Руководство по развертыванию и использованию программного средства.	37
Заключение	50
Список использованных источников	51
Приложение А (обязательное) Отчет о проверке на заимствование в системе «Антиплагиат».....	52
Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику	53
Приложение В (обязательное) Листинг скрипта генерации базы данных.....	63

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ

БД (база данных)	– представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины
Информационная система	– система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые), которые обеспечивают и распространяют информацию
Нормальная форма	– свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных
Среда выполнения	– вычислительное окружение, необходимое для выполнения компьютерной программы и доступное во время выполнения компьютерной программы
СУБД (система управления базами данных)	– совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных
ТЗ (техническое задание)	– документ, содержащий требования заказчика к объекту разработки, определяющий порядок и условия её проведения
<i>API (application programming interface)</i>	– описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой
<i>IDE (Integrated development environment)</i>	– комплекс программных средств, используемый программистами для разработки программного обеспечения
<i>IDEF</i>	– методология функционального моделирования (англ. <i>function modeling</i>) и графическая нотация, предназначенная для формализации и описания бизнес-процессов
<i>Java</i>	– строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией <i>Sun Microsystems</i>
<i>SQL (structured query language)</i>	– язык структурированных запросов, декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной

<i>MySQL Server</i>	<p>базе данных, управляемой соответствующей системой управления базами данных</p> <p>– свободная реляционная система управления базами данных</p>
<i>UML (Unified Modeling Language)</i>	<p>– язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур</p>
<i>Декомпозиция</i>	<p>– разделение сложного объекта, системы, задачи на составные части, элементы. Она показывает из каких более мелких работ состоит основной процесс.</p>

и законы. В качестве механизма осуществления главной функции выступают, администратор, который отвечает за изменение каталога, управление пользователями, контролем функционирования магазина, графический-интерфейс для осуществления связи клиента и магазина. Выходным параметром для данной системы являются электронный чек и отправка товара.

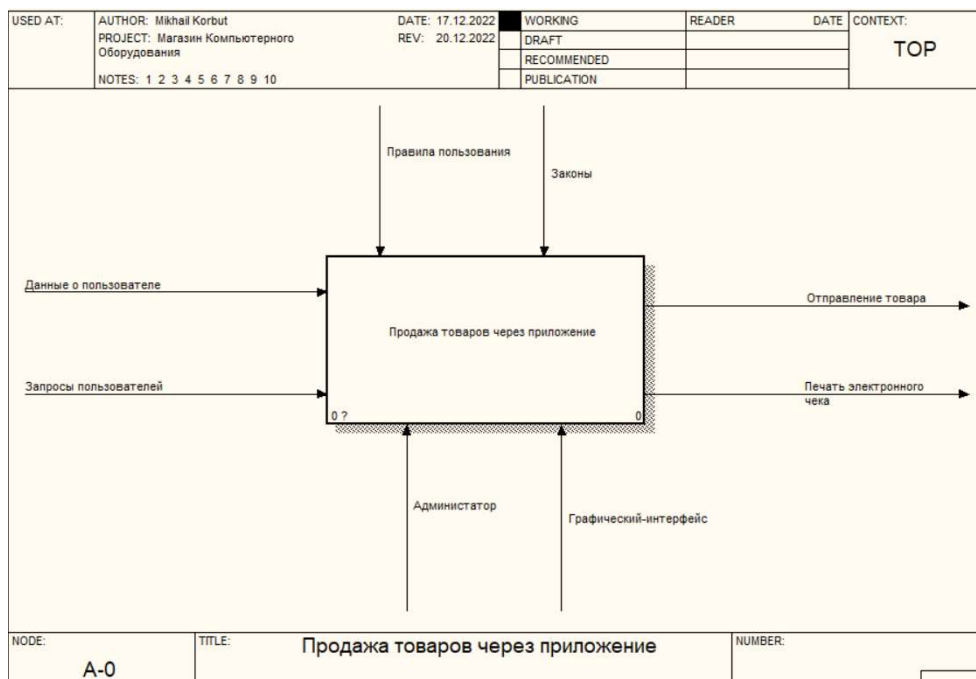


Рисунок 1.1 – Контекстный уровень диаграммы.

Далее представлена декомпозиция контекстной диаграммы, состоящая из трех блоков (рисунок 1.2):

1. Регистрация пользователя.
2. Добавление товаров в корзину.
3. Подтверждение заказа.

Декомпозиция – это разделение сложного объекта, системы, задачи на составные части, элементы. Она показывает из каких более мелких работ состоит основной процесс.

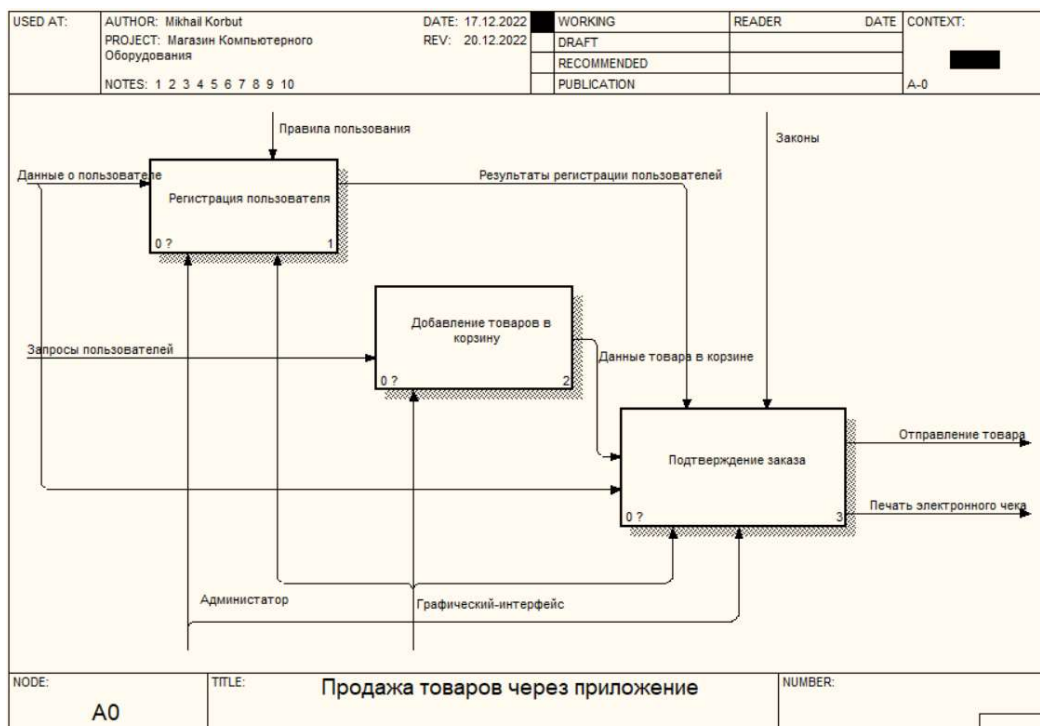


Рисунок 1.2 – Декомпозиция контекстной диаграммы.

Этап «Добавление товаров в корзину» разбит на два функциональных блока (рисунок 1.3):

1. Проверка наличия товара на складе.
2. Отправка данных о товаре в корзину.

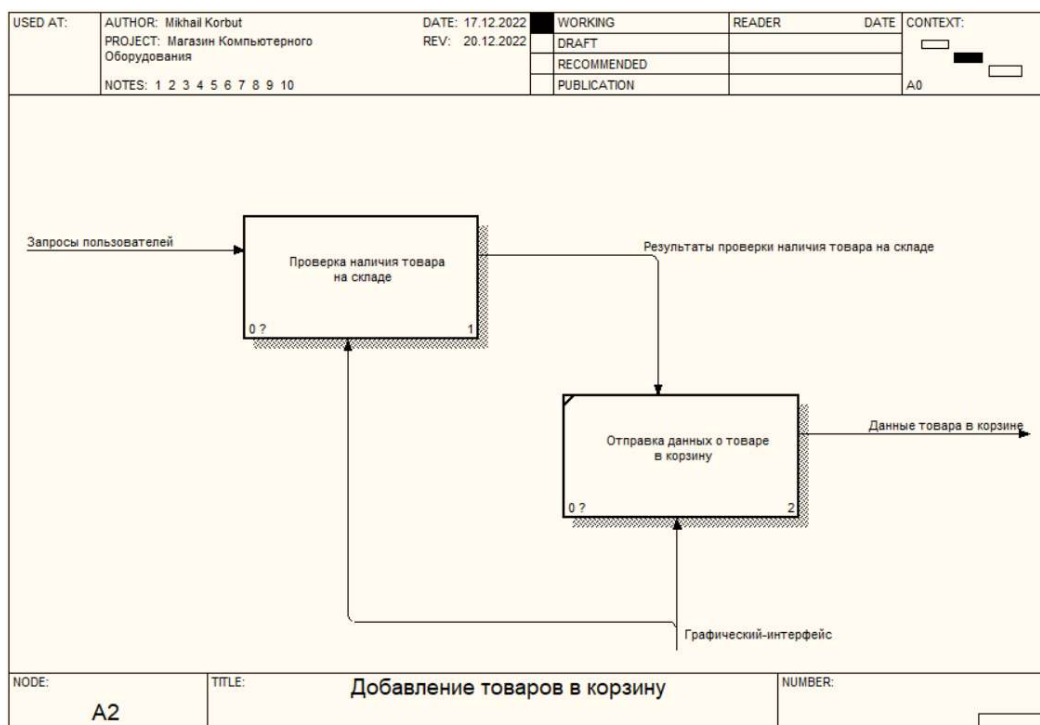


Рисунок 1.3 – Декомпозиция блока «Добавление товаров в корзину».

Этап «Проверка наличия товара на складе» разбит на два функциональных блока (рисунок 1.4):

1. Запрос информации о товаре на складе.
2. Анализ результатов запроса товара

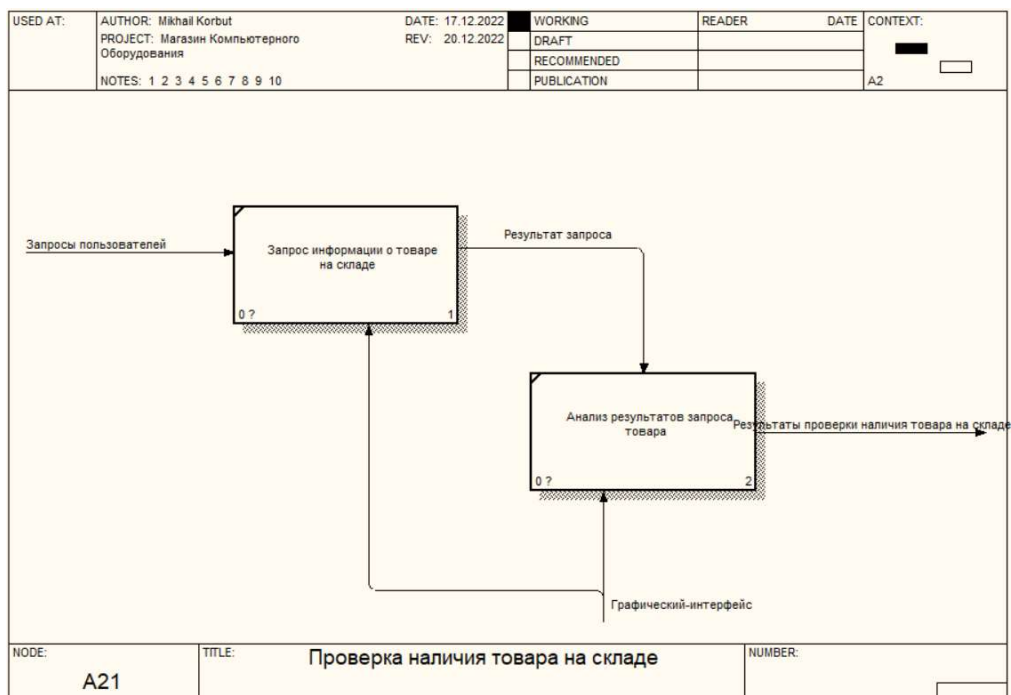


Рисунок 1.4 – Декомпозиция блока «Проверка наличия товара на складе».

Этап «Подтверждение заказа» разбит на три функциональных блока (рисунок 1.5):

1. Запрос информации о товаре на складе.
2. Подтверждение заказа пользователя.
3. Подтверждение заказа.

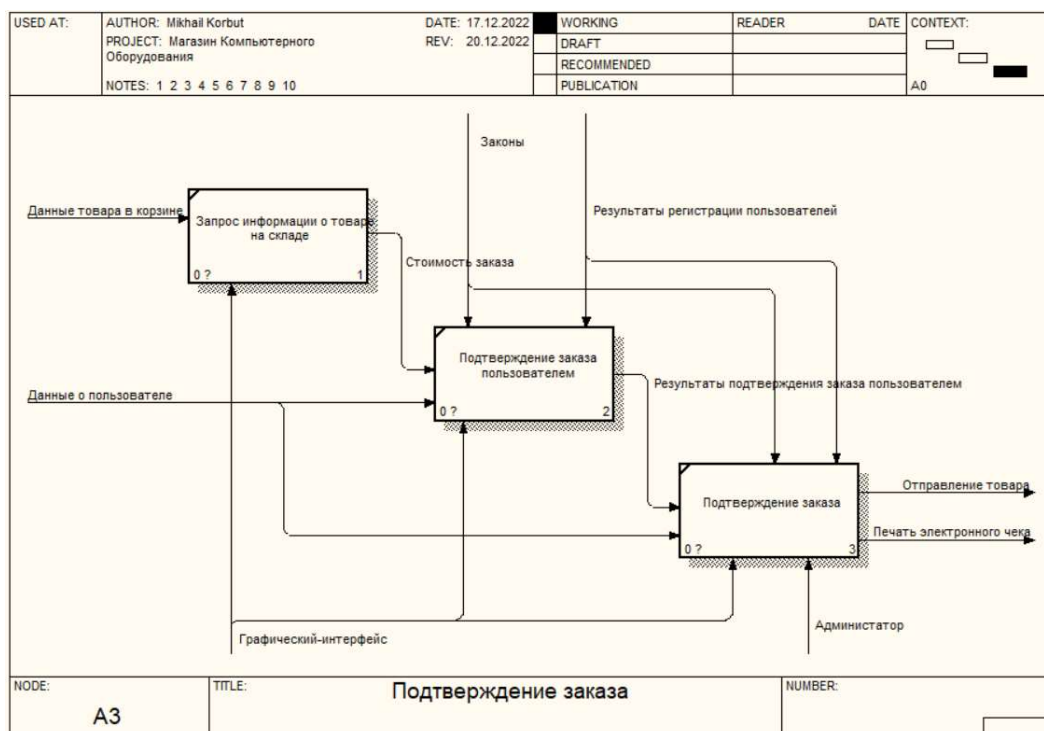


Рисунок 1.5 – Декомпозиция блока «Подтверждение заказа».

1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. Каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

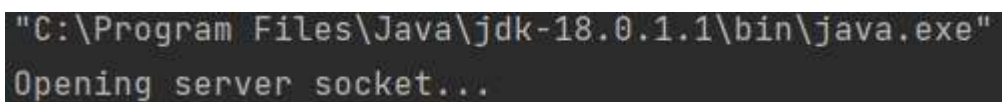
Оба актера могут зайти под своей ролью и, в зависимости от авторизации, им даются разные возможности (рисунок 1.6).

4 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

Приложение является клиент-серверным и функционирует с базой данных MySQL. Для входа предусмотрены 2 роли: администратор и пользователь.

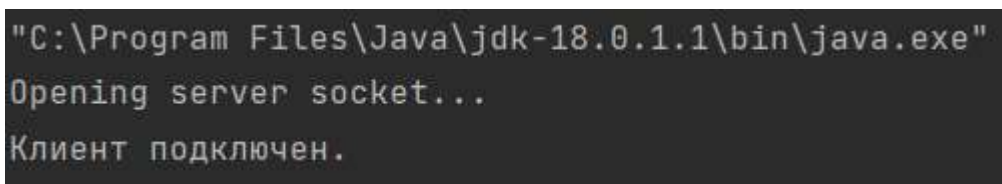
Для запуска серверной части необходимо установить jdk 18. А также обязательным является установка MySQL Workbench. Необходимо последовательно запустить server.jar, а затем client.jar и скрипт для создания базы данных.

Для запуска сервера необходимо в командной строке ввести команду `java -jar Server.jar` предварительно переместившись в каталог, в котором находится jar-архив. После чего можно запустить графическое приложение для работы клиента. Пример запуска показан на рисунке 4.1. Пример отображения состояния подключений показан на рисунке 4.2.



```
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.exe"  
Opening server socket...
```

Рисунок 4.1 – Запуск сервера.



```
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.exe"  
Opening server socket...  
Клиент подключен.
```

Рисунок 4.2 – Отображение состояний подключения к серверу.

При запуске клиентской части отображается меню входа и регистрации. Соответственно есть возможность пройти процесс авторизации как пользователь или как администратор, или если нет аккаунта пользователя, то зарегистрироваться (рисунок 4.3)

Оформим заказ для этого нажмем на кнопку «Оформить заказ» (рисунок 4.11).

Оформление заказа

ФИО: Иван Годин
Номер телефона: +375 44 670 1244
Электронная почта: goodin33@gmail.com
Почтовый индекс: 2201525
Адрес: ул. Высокая 15, кв. 120

Список товаров

Название	Категория	Стоимость
Lenovo Legion	Ноутбуки	700.99\$
Asus VivoBook 15s	Ноутбуки	550\$
MSI Gaming Pro	Ноутбуки	890\$

Итого к оплате: 2140.99\$

← Оформить

Рисунок 4.11 – Просмотр корзины.

Вернемся в меню пользователя и посмотрим историю покупок клиента пункт меню «История покупок» (рисунок 4.12).

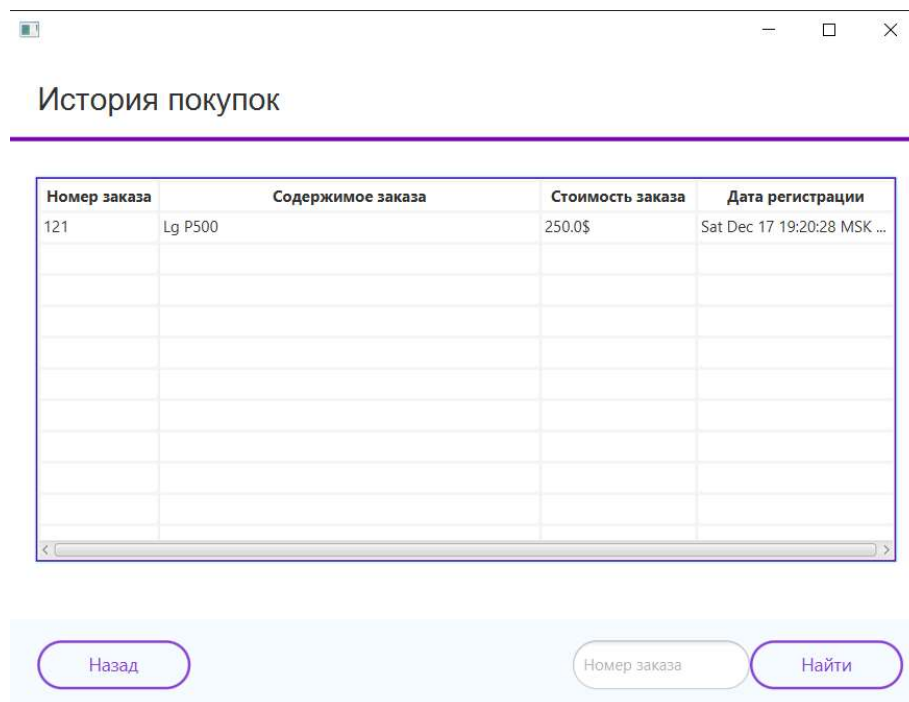


Рисунок 4.12– Просмотр корзины.

После описания всех функций пользователя перейдем к меню админа (рисунок 4.13). Так же, как и в меню пользователя после нажатия на иконку администратора откроется окно с информацией о пользователе. Рассмотрим первый пункт меню «Работа с клиентами» (рисунок 4.14).

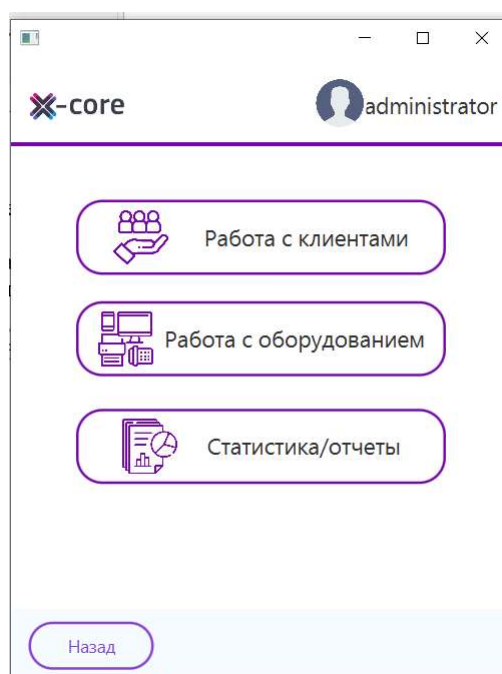


Рисунок 4.13– Меню администратора.

ПРИЛОЖЕНИЕ А
(обязательное)
Отчет о проверке на заимствование в системе «Антиплагиат»

The screenshot displays the 'Антиплагиат' (AntiPlagiat) system interface. At the top, the header includes the system logo, user information (korbut2792@gmail.com), and navigation links. The main content area shows the results of a document check:

- Оригинальность** (Originality): 92.49%
- Заимствования** (Plagiarism): 7.51%
- Цитирования** (Citations): 0%
- Самоцитирования** (Self-citations): 0%

Below the results, there are buttons for 'ПОЛНЫЙ ОТЧЕТ' (Full Report), 'КРАТКИЙ ОТЧЕТ' (Short Report), and 'ИСТОРИЯ ОТЧЕТОВ' (Report History). Action buttons include 'РАСПЕЧАТАТЬ' (Print), 'ВЫГРУЗИТЬ' (Download), and 'СОЗДАТЬ ССЫЛКУ' (Create Link).

The 'Свойства документа' (Document Properties) section on the left lists options: 'Свойства документа', 'Структура документа', 'Текстовые метрики' (Text Metrics), and 'Параметры проверки' (Check Parameters). The 'Текстовые метрики' option is highlighted with a 'NEW' tag.

The 'Свойства документа' section on the right contains the following fields:

- Имя исходного файла** (Original file name): отчет.pdf
- Авторы документа** (Document authors): Корбут, Михаил
- Название документа** (Document title): отчет
- Тип документа** (Document type): Не указано

A 'РЕДАКТИРОВАТЬ СВОЙСТВА' (Edit properties) button is located at the bottom of this section.

Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

```
// Класс MenuAdminController показывает основные принципы работы контроллера.  
package controllers;
```

```
public class MenuAdminController {  
  
    @FXML  
    private ResourceBundle resources;  
  
    @FXML  
    private URL location;  
  
    @FXML  
    private Button backButton;  
  
    @FXML  
    private Button dbWorkButton;  
  
    @FXML  
    private Button clientWorkButton;  
  
    @FXML  
    private Button statisticWorkButton;  
  
    @FXML  
    private Button personalInfButton;  
  
    @FXML  
    void statisticWork(ActionEvent event) throws IOException {  
        statisticWorkButton.getScene().getWindow().hide();  
  
        FXMLLoader loader = new FXMLLoader();  
        loader.setLocation(getClass().getResource("/statisticWork.fxml"));  
  
        try {  
            loader.load();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
  
        Parent root = loader.getRoot();  
        Stage stage = new Stage();  
        stage.setScene(new Scene((root)));  
        stage.show();  
    }  
  
    @FXML  
    void persInf(ActionEvent event) throws IOException {  
        Connect.client.sendMessage("adminInf");  
        WindowChanger.changeWindow(getClass(), personalInfButton,  
        "adminInformation.fxml", "", false);  
    }  
  
    @FXML  
    void backToMain(ActionEvent event) {  
        backButton.getScene().getWindow().hide();  
  
        FXMLLoader loader = new FXMLLoader();  
        loader.setLocation(getClass().getResource("/main.fxml"));  
  
        try {
```

```

        loader.load();
    } catch (IOException e) {
        e.printStackTrace();
    }

    Parent root = loader.getRoot();
    Stage stage = new Stage();
    stage.setScene(new Scene((root)));
    stage.show();
}
}
// Методы initialize и getCategory() отвечающие за добавление записей в xml
таблицу. Метод initialize переопределяет метод интерфейса Initializable.
@Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        nameColumn.setCellValueFactory(field -> new
SimpleObjectProperty<>(field.getValue().getName()));
        parameter1Column.setCellValueFactory(field -> new
SimpleObjectProperty<>(field.getValue().getParameter_1()));
        parameter2Column.setCellValueFactory(field -> new
SimpleObjectProperty<>(field.getValue().getParameter_2()));
        categoryTable.setItems(getCategory());
    }

    private ObservableList<Category> getCategory() {
        ObservableList<Category> categoryList =
FXCollections.observableArrayList();
        ArrayList<Category> category = (ArrayList<Category>)
Connect.client.readObject();
        System.out.println(category);
        categoryList.addAll(category);
        categoryTable.setItems(categoryList);
        return categoryList;
    }

// Метод реализующий смену окон.
public static void changeWindow(Class className, Button button, String fname,
String title, boolean ismodal) throws IOException {
    FXMLLoader fxmlloader = new FXMLLoader();
    fxmlloader.setLocation(className.getResource("/" + fname));
    fxmlloader.load();
    Parent root = fxmlloader.getRoot();
    Stage stage = new Stage();
    stage.setTitle(title);
    stage.setScene(new Scene(root));
    if (ismodal) {
        stage.initModality(Modality.APPLICATION_MODAL);
    }
    else {
        button.getScene().getWindow().hide();
    }
    stage.show();
}

// Класс Check, методы этого класса проверяют вводимые значения на
соответствие запрашиваемому типу данных.

public class Check {
    public static boolean checkInt(String str) {
        Pattern r = Pattern.compile("[\\d]+");
        Matcher m = r.matcher(str);
        return m.matches();
    }
}

```

```

    }

    public static boolean checkDouble(String str) {
        Pattern r = Pattern.compile("[+-]?([0-9]*[.])?[0-9]+");
        Matcher m = r.matcher(str);
        return m.matches();
    }

    public static boolean checkString(String str) {
        Pattern r = Pattern.compile("[a-zA-Z]+");
        Matcher m = r.matcher(str);
        return m.matches();
    }
}

// Knacc Client.

public class Client {
    private Socket clientSocket;
    private ObjectOutputStream outputStream;
    private ObjectInputStream inputStream;

    private String message;

    public Client(String ipAddress, String port){
        try {
            clientSocket = new Socket(ipAddress, Integer.parseInt(port));
            outputStream = new
ObjectOutputStream(clientSocket.getOutputStream());
            inputStream = new ObjectInputStream(clientSocket.getInputStream());
        } catch (IOException e) {
            System.out.println("Server not found: " + e.getMessage());
            System.exit(0);
        }
    }

    public void sendMessage(String message){
        try {
            outputStream.writeObject(message);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void sendObject(Object object){
        try {
            outputStream.writeObject(object);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public String readMessage() throws IOException {
        try {
            message = (String) inputStream.readObject();
        } catch (ClassNotFoundException | IOException e) {
            e.printStackTrace();
        }

        return message;
    }
}

```

```

public Object readObject(){
    Object object = new Object();
    try {
        object = inStream.readObject();
    } catch (ClassNotFoundException | IOException e) {

        e.printStackTrace();
    }
    return object;
}

public void close() {
    try {
        clientSocket.close();
        //outStream.flush();
        inStream.close();
        outStream.close();
    } catch (EOFException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

ПРИЛОЖЕНИЕ В (ОБЯЗАТЕЛЬНОЕ) ЛИСТИНГ СКРИПТА ГЕНЕРАЦИИ БАЗЫ ДАННЫХ

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE
,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----
-- Schema mydb
-- -----
-- -----
-- Schema computerequipmentstore
-- -----

CREATE SCHEMA IF NOT EXISTS `foreignlanguageschool` DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `computerequipmentstore` ;

CREATE TABLE `admins` (
  `idadmins` int NOT NULL AUTO_INCREMENT,
  `id_keys` int NOT NULL,
  PRIMARY KEY (`idadmins`),
  UNIQUE KEY `id_keys_UNIQUE` (`id_keys`),
  CONSTRAINT `fk_admins_keys` FOREIGN KEY (`id_keys`) REFERENCES `keys`
(`id_keys`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `basket` (
  `idequipment` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `producer` varchar(45) NOT NULL,
  `category` varchar(45) NOT NULL,
  `parameter1` varchar(45) NOT NULL,
  `parameter2` varchar(45) NOT NULL,
  `price` varchar(45) NOT NULL,
  PRIMARY KEY (`idequipment`),
  UNIQUE KEY `name_UNIQUE` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=146 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `categories` (
  `idcategory` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `parameter1` varchar(45) NOT NULL,
  `parameter2` varchar(45) NOT NULL,
  PRIMARY KEY (`idcategory`),
  UNIQUE KEY `name_UNIQUE` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=93 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `client` (
  `idclient` int NOT NULL AUTO_INCREMENT,
  `firstname` varchar(45) NOT NULL,
  `lastname` varchar(45) NOT NULL,
```



```

    `id_keys` int NOT NULL,
    `orders_amount` int DEFAULT '0',
    `total_spent` double DEFAULT '0',
    PRIMARY KEY (`idclient`),
    UNIQUE KEY `id_keys_UNIQUE` (`id_keys`),
    KEY `fk_clients_keys_idx` (`id_keys`),
    CONSTRAINT `fk_clients_keys` FOREIGN KEY (`id_keys`) REFERENCES `keys`
    (`id_keys`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=47 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `equipment` (
  `idequipment` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `producer` varchar(45) NOT NULL,
  `category` varchar(45) NOT NULL,
  `firstParameter` varchar(45) NOT NULL,
  `secondParameter` varchar(45) NOT NULL,
  `price` varchar(45) NOT NULL,
  PRIMARY KEY (`idequipment`),
  UNIQUE KEY `name_UNIQUE` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `keys` (
  `id_keys` int NOT NULL AUTO_INCREMENT,
  `login` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  PRIMARY KEY (`id_keys`),
  UNIQUE KEY `login_UNIQUE` (`login`)
) ENGINE=InnoDB AUTO_INCREMENT=78 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `orders` (
  `idorder` int NOT NULL AUTO_INCREMENT,
  `iduser` int NOT NULL,
  `contents` varchar(45) NOT NULL,
  `sumprice` double DEFAULT NULL,
  `date` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idorder`)
) ENGINE=InnoDB AUTO_INCREMENT=122 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `delete_basket`()
BEGIN
    DELETE FROM basket;
END$$
DELIMITER ;

```

```

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `delete_category`(name_
varchar(45))
BEGIN
    delete from `categories`
    where name = name_;
    delete from `equipment`
    where category = name_;
END$$
DELIMITER ;

DELIMITER $$

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete_client`(login_
varchar(45))
BEGIN
    select `id_keys` into @id_k from `keys` where `login` = login_;
    delete from `keys`
    where id_keys = @id_k;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `delete_equipment`(name_
varchar(45))
BEGIN
    delete from `equipment`
    where name = name_;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`delete_EquipmnetFromBasket`(name_ varchar(45))
BEGIN
    delete from `basket`
    where name = name_;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `find_login`(uslogin VARCHAR(45),
uspass VARCHAR(45), OUT id_user INT, OUT usrole VARCHAR(10))
BEGIN
    SET id_user = 0;
    SET usrole = "";
    SELECT id_keys INTO id_user FROM `keys`
    WHERE `login` = uslogin AND `password` = uspass;

    SELECT COALESCE(ur, "") into usrole
    FROM ( select "client" as ur from `client` where id_keys = id_user
    union
    select "admin" as ur from `admins` where id_keys = id_user
    ) as T;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_admin`()
BEGIN
    select `keys`.login, `keys`.`password` from admins
    join `keys` on `keys`.id_keys = admins.id_keys;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_admin`(login varchar(45),
pass varchar(45))
BEGIN
    CALL insert_keys(login, pass, @id_keys);
    INSERT INTO `computerequipmentstore`.`admins` (`id_keys`)
    VALUES (@id_keys);
END$$
DELIMITER ;

```

```

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_equipment`(name
varchar(45), producer varchar(45), category varchar(45), parameter1
varchar(45), parameter2 varchar(45), price varchar(45))
BEGIN
    insert into `computerequipmentstore`.`equipment` (`name`, `producer`,
`category`, `firstParameter`, `secondParameter`, price)
        VALUES (name, producer, category, parameter1, parameter2, price);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `make_order`(iduser int, contents
varchar(45), sumprice double, dat varchar(45))
BEGIN
    insert into `computerequipmentstore`.`orders` (iduser, contents,
sumprice, date)
        VALUES (iduser, contents, sumprice, dat);
    UPDATE `computerequipmentstore`.`client`
    SET `total_spent` = sumprice + total_spent
    WHERE id_keys = iduser;
    UPDATE `computerequipmentstore`.`client`
    SET `orders_amount` = orders_amount + 1
    WHERE id_keys = iduser;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_keys`(login varchar(45),
pass varchar(45), out id_keys int)
BEGIN
    insert into `computerequipmentstore`.`keys`(`login`, `password`)
        values (login, pass);
    select last_insert_id() into id_keys;
END$$
DELIMITER ;

```