МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННО-СЕТЕВЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | Е.О. Пятлина |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| Объектно-ориентированное проектирование программного обеспечения ИНФОРМАЦИОННОЙ СИСТЕМЫ “система автодоната игровых серверов” |
| по дисциплине: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4632 |  |  |  | C.C. Шульгин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

Оглавление

[ВВЕДЕНИЕ. 4](#_Toc513119132)

[1. Описание основных функций Информационной системы «Система автодоната игровых серверов» 4](#_Toc513119133)

[2. Краткая информация об аппарате проектирования. Язык UML. 4](#_Toc513119134)

[3. CASE средства, используемые для проектирования ИС. 4](#_Toc513119136)

[4. Разработка ПО информационной системы «Система автодоната игровых серверов» 4](#_Toc513119137)

[4.1 Use-case диаграмма (диаграмма вариантов использования, сценариев, прецедентов). 5](#_Toc513119138)

[4.2 Диаграмма классов 7](#_Toc513119139)

[4.3 Диаграммы последовательностей 9](#_Toc513119140)

[4.4 Диаграммы состояний 13](#_Toc513119141)

[4.5 Диаграммы видов деятельности 20](#_Toc513119142)

[4.6 Диаграмма размещения. 25](#_Toc513119143)

[4.7 Диаграмма пакетов 25](#_Toc513119144)

[ЗАКЛЮЧЕНИЕ 26](#_Toc513119147)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc513119148)

[ПРИЛОЖЕНИЕ 27](#_Toc513119149)

# ВВЕДЕНИЕ. В данной работе приведен проект построения структуры программного обеспечения информационной системы. В качестве предметной области рассмотрена «Система автодоната игровых серверов». Задача данной системы: обеспечить возможность пользователям пополнять счет и совершать покупки внутри-игровых элементов через сайт игры, а так же позволять администрации изменять содержимое страниц магазина через админ-панель. В ходе данной работы была произведена автоматическая генерация кода программной реализации системы, используя диаграмму классов.

# 1. Описание основных функций Информационной системы «Система автодоната игровых серверов»

* Пополнение баланса пользователя
  + Переадресация на сайт платежного агрегатора
  + Обеспечение связи с платежным агрегатором
* Отображение содержимого на странице сайта
  + Отображение баланса пользователя
  + Отображение списка донат-страниц
  + Отображение содержимого донат-страницы
  + Отображение категорий
  + Отображение товаров
* Возможность приобретения товаров с выбором игрового сервера
* Возможность редактировать содержимое администратором сайта
  + Редактирование донат-страниц
  + Редактирование категорий
  + Редактирование товаров
* Возможность управлять балансами пользователей администратором сайта
* Обеспечение связи с игровыми серверами для передачи информации о совершенных покупках.

# 2. Краткая информация об аппарате проектирования. Язык UML.

# UML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

# 3. CASE средства, используемые для проектирования ИС.

В качестве CASE средства, используемого для проектирования данной информационной системы, использовалось специальное программное обеспечение Sofrware Ideas Modeler. Данная программа позволяет работать со всеми видами диаграмм языка UML, производить генерацию кода на множестве языков программирования, а так же имеет еще множество других функций. Среди недостатков данной программы – в некоторых местах она дает сбои, что немного мешает в ходе работы.

# 4. Разработка ПО информационной системы «Система автодоната игровых серверов»

По требованиям к данной работе нам нужно производить расчет оценки диаграмм. Для этого введем необходимые формулы:

**Sdiagr=** (формула 1) **,**

где Sobj-оценка элемента на диаграмме, Slink- оценка связей, Оbj- кол-во объектов на диаграмме, Tobj –количество типов объектов, Tlink- количество типов связи.

(формула 2),

где Op – число операций класса, Atr – число атрибутов класса

## **4.1 Use-case диаграмма (диаграмма вариантов использования, сценариев, прецедентов).**

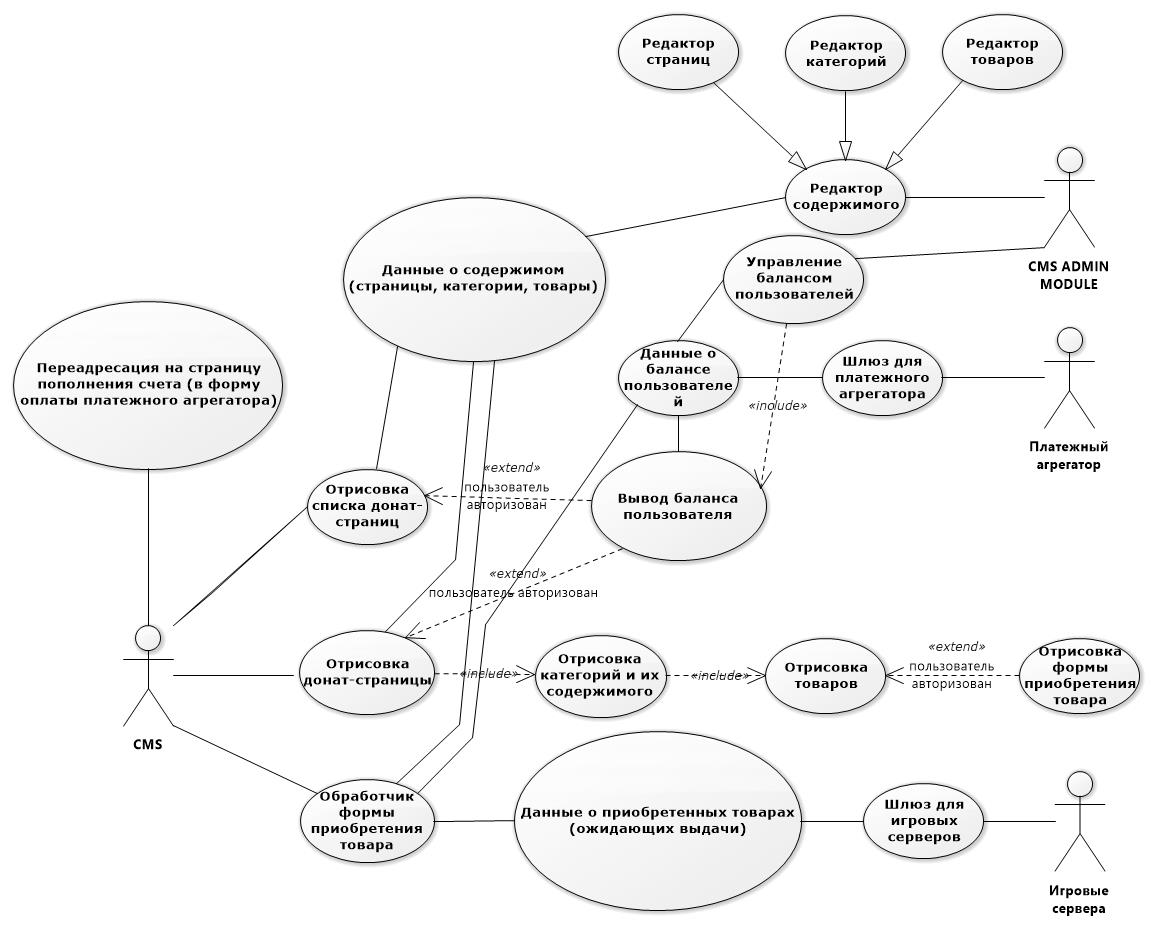


Рисунок 1 – Диаграмма вариантов использования

На данной диаграмме изображена общий функционал системы. Взглянув на эту диаграмму можно понять, для чего предназначена наша система и как в общем она устроена.

**Расчет оценки диаграммы**

Sdiagr = = 3.26 (по формуле 1)

Значение немного выше рекомендованного, близка к указанному диапазону.

## **4.2 Диаграмма классов**

Диаграмма классов — статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами

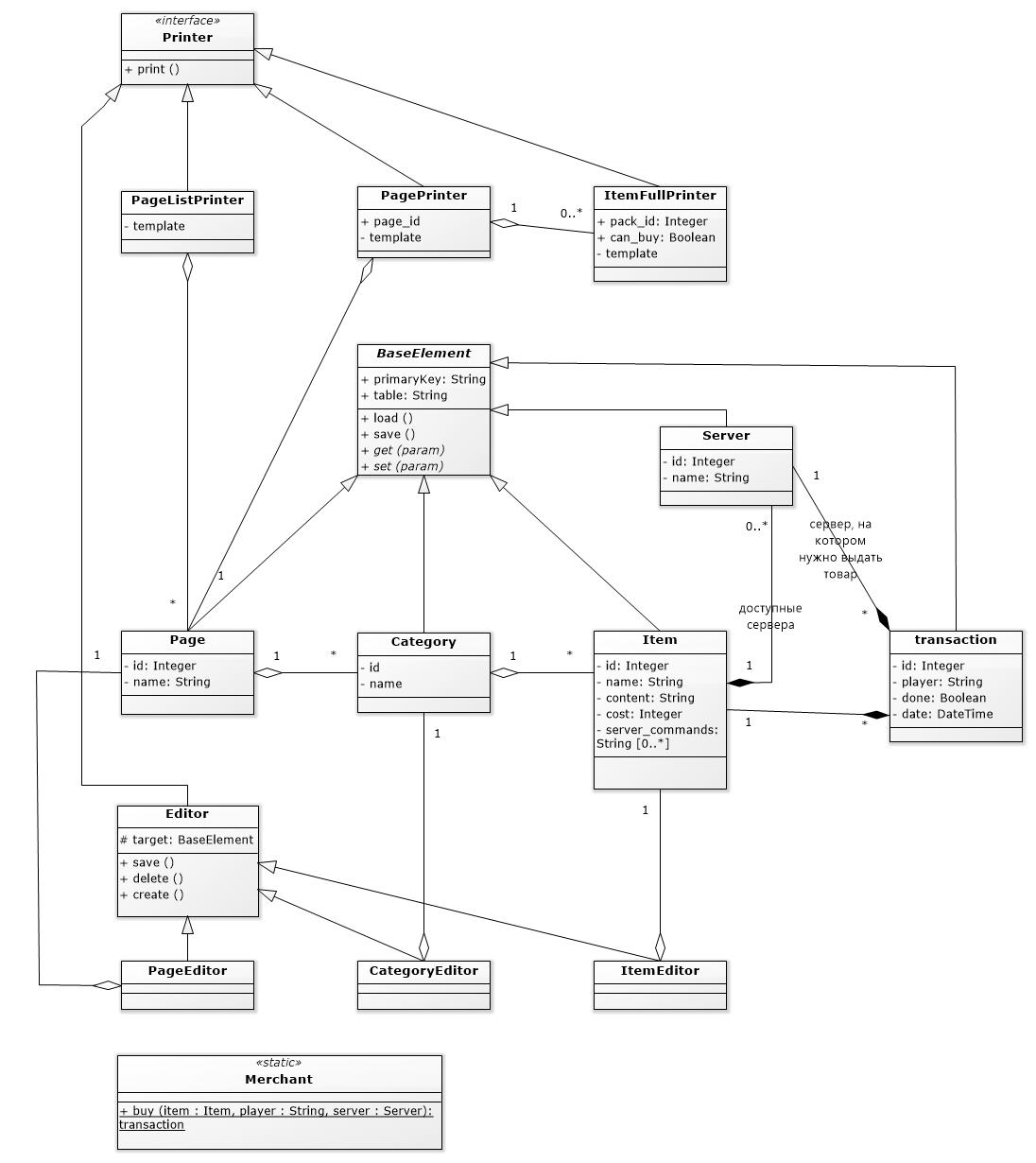


Рисунок 2 – Диаграмма классов

На данной диаграмме изображены основные классы, которые должны присутствовать в системе, их иерархия и взаимосвязь. Имея данную диаграмму можно сгенерировать основу программной реализации системы, проведя генерацию кода. В приложении к данной работе размещен код на языке PHP, сгенерированный по данной модели при помощи Sofrware Ideas Modeler.

**Расчет оценки диаграммы**

Расчет для классов (по формуле 2)

Scls(Printer) = = 3.33

Scls(PageListPrinter) = = 3.33

Scls(PagePrinter) = = 2.36

Scls(ItemFullPrinter) = = 1.92

Scls(BaseElement) = = 1.90

Scls(Server) = = 2.36

Scls(Page) = = 2.36

Scls(Category) = = 2.36

Scls(Item) = = 1.49

Scls(transaction) = = 1.67

Scls(Editor) = = 2.28

Scls(Merchant) = = 3.33

∑Scls = 23.97

Sdiagr = = 8.55 (по формуле 1)

Значение получилось выше рекомендованного. Это показывает, что информация в нашей диаграмме избыточна.

## **4.3 Диаграммы последовательностей**

Диаграмма последовательностей — диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления. Основными элементами диаграммы последовательностей являются обозначения объектов (прямоугольники), вертикальные линии (линии жизни), отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами.

Объекты - это информационные единицы, участвующие в сценарии. Первым указывается объект, который является инициатором данного сценария. Объекты обмениваются сообщениями, порядок их указывается.

На данной диаграмме объекты располагаются слева направо.

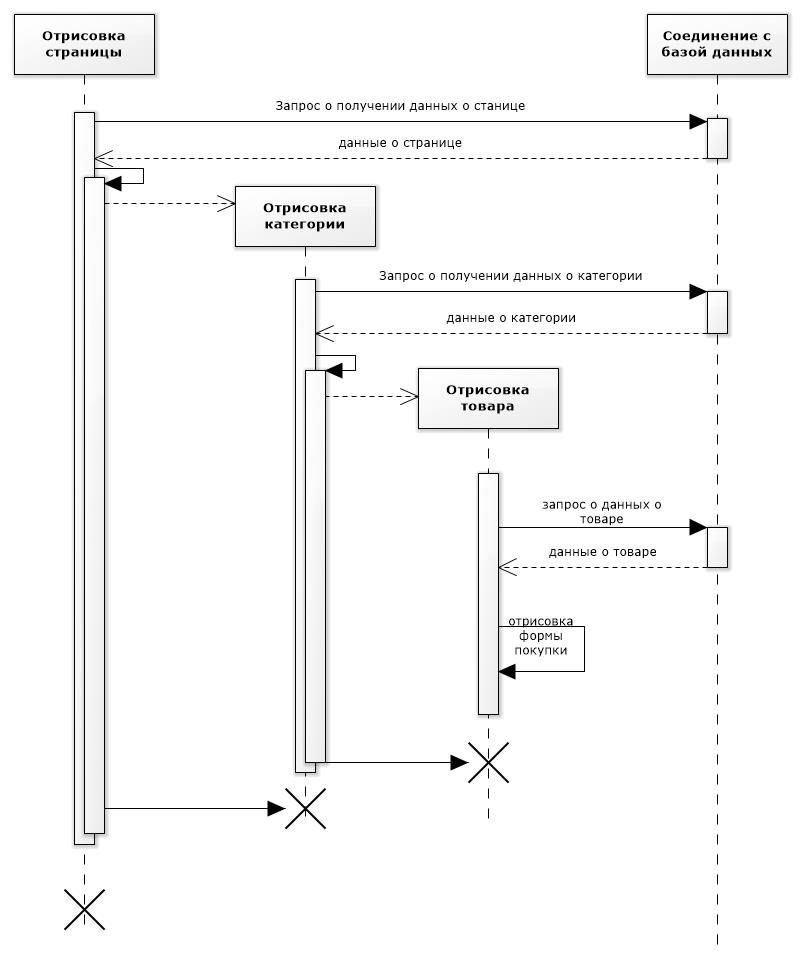
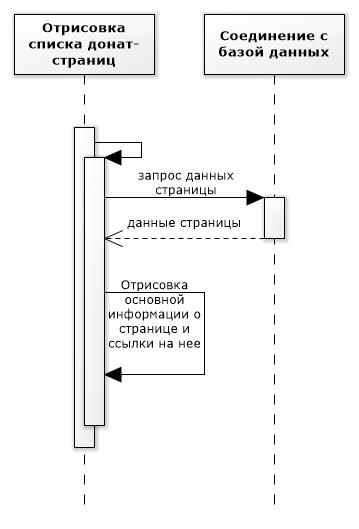


Рисунок 3 – диаграмма последовательностей для сценария “Отрисовка страницы”

**Расчет оценки диаграммы**

Sdiagr = = 5.14 (по формуле 1)

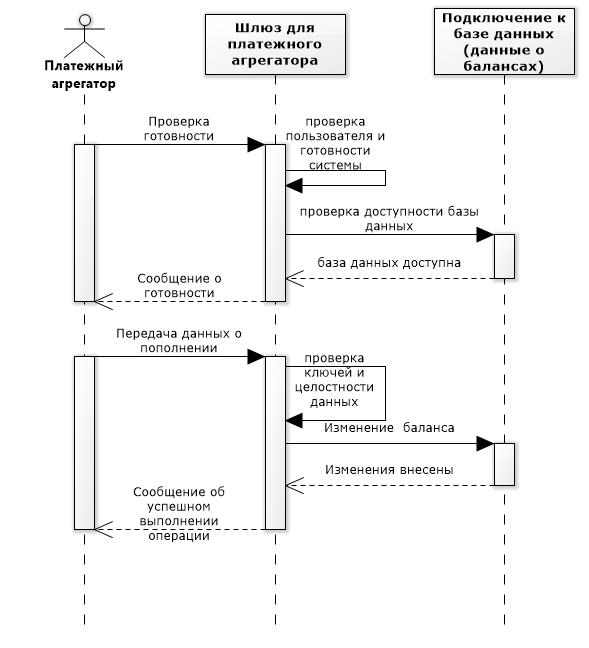
Рисунок 4 – диаграмма последовательностей для сценария “Отрисовка списка страниц”



**Расчет оценки диаграммы**

Sdiagr = = 3.17 (по формуле 1)

Рисунок 5 – диаграмма последовательностей для варианта использования “Шлюз для платежного агрегатора”



**Расчет оценки диаграммы**

Sdiagr = = 3.83 (по формуле 1)

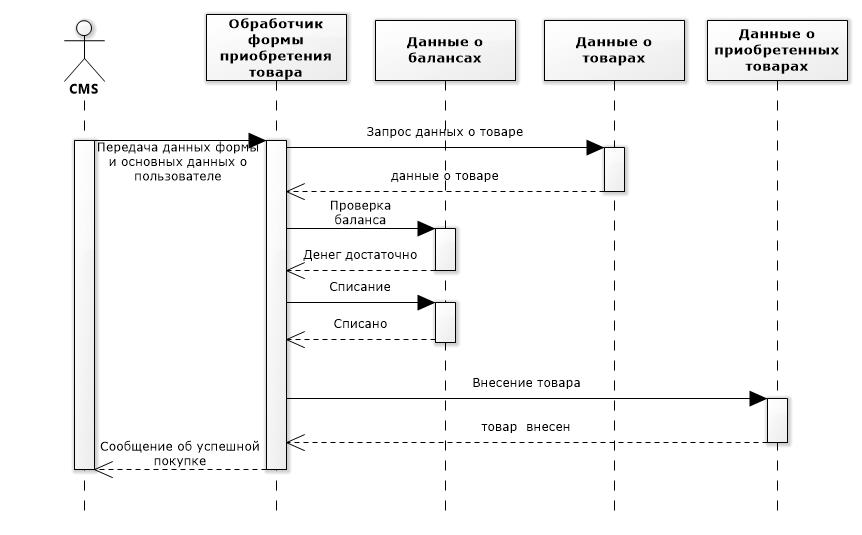


Рисунок 6 – диаграмма последовательностей для сценария “Обработка формы приобретения товара”

**Расчет оценки диаграммы**

Sdiagr = = 4.14 (по формуле 1)

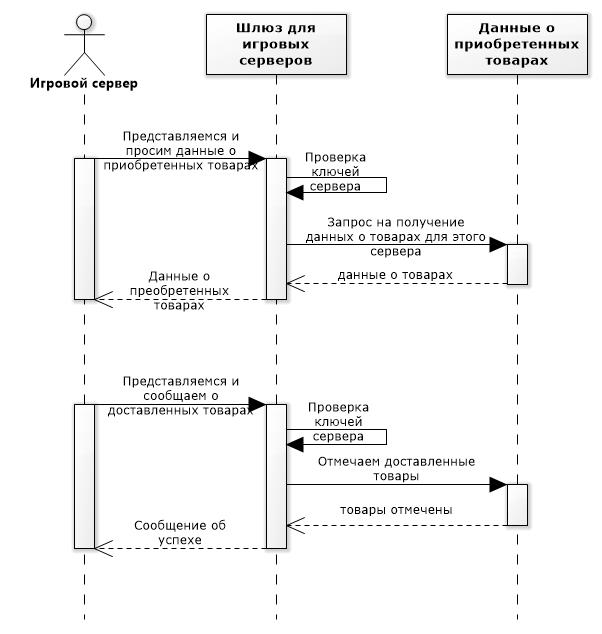


Рисунок 7 – диаграмма последовательностей для варианта использования “Шлюз для игровых серверов”

**Расчет оценки диаграммы**

Sdiagr = = 3.83 (по формуле 1)

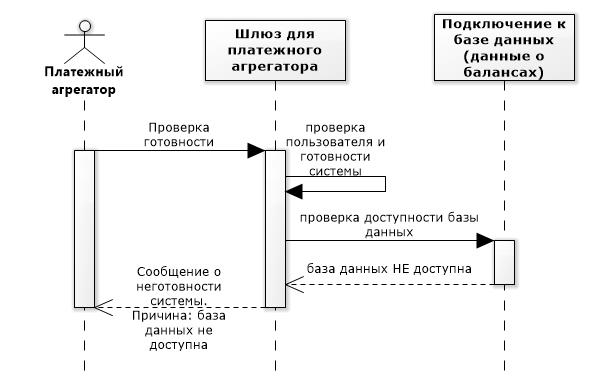


Рисунок 8 – диаграмма последовательностей для варианта использования “Шлюз для платежного агрегатора” при условии недоступности базы данных

**Расчет оценки диаграммы**

Sdiagr = = 2.97 (по формуле 1)

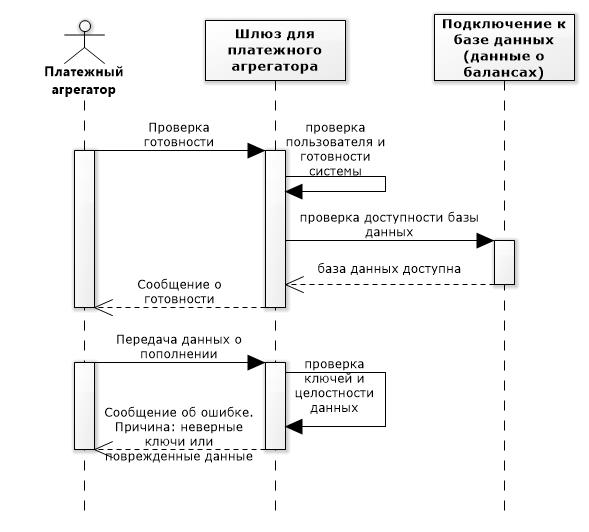


Рисунок 9 – диаграмма последовательностей для варианта использования “Шлюз для платежного агрегатора” при условии передачи агрегатором неверных ключей или поврежденных данных

Sdiagr = = 3.48 (по формуле 1)

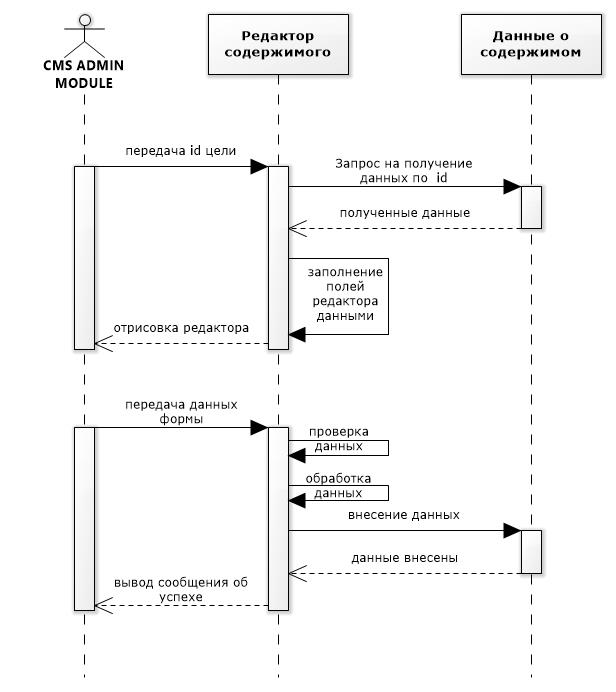


Рисунок 10 – диаграмма последовательностей для варианта использования “Редактор содержимого”

Варианты использования “Редактор категорий”, “Редактор страниц” и “Редактор товаров” связаны обобщением с вариантом использования “Редактор содержимого” и имеют аналогичную диаграмму последовательностей. Различие между ними только в типе информации, над которой они работают.

**Расчет оценки диаграммы**

Sdiagr = = 4.01 (по формуле 1)

## **4.4 Диаграммы состояний**

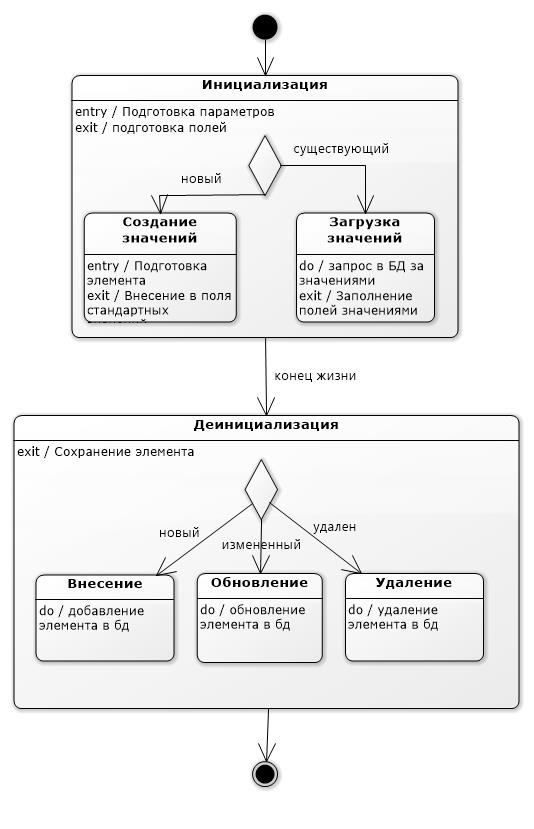


Рисунок 11 – Диаграмма состояний для класса BaseElement и всех его потомков

Все потомки этого класса работают по тому же типу, изменяется только тип данных, над которыми они работают

**Расчет оценки диаграммы**

Sdiagr = (по формуле 1)

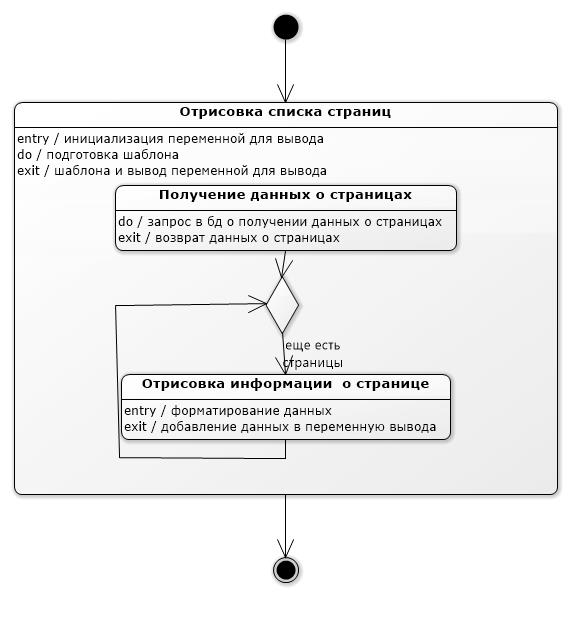


Рисунок 12 – Диаграмма состояний для класса PageListPrinter

**Расчет оценки диаграммы**

Sdiagr = (по формуле 1)

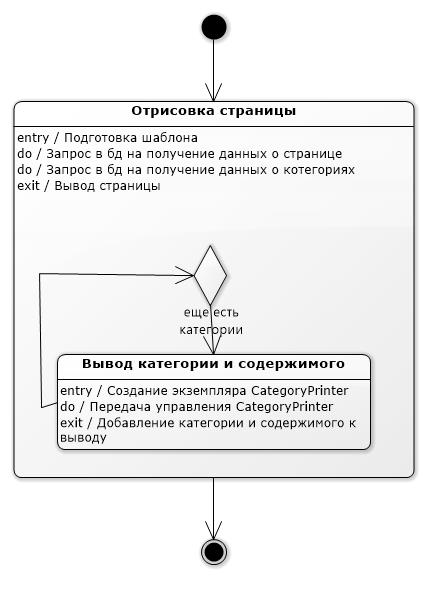


Рисунок 13 – Диаграмма состояний для класса PagePrinter

**Расчет оценки диаграммы**

Sdiagr = (по формуле 1)

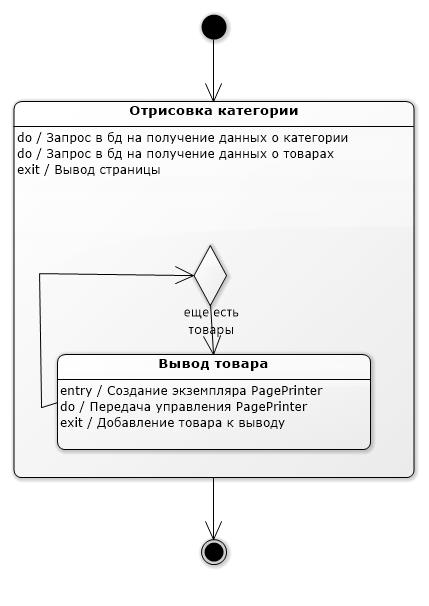


Рисунок 14 – Диаграмма состояний для класса CategoryPrinter

**Расчет оценки диаграммы**

Sdiagr = (по формуле 1)

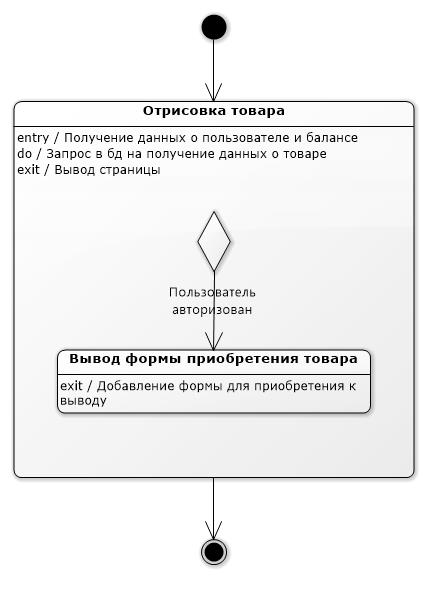


Рисунок 15 – Диаграмма состояний для класса ItemPrinter

**Расчет оценки диаграммы**

Sdiagr = (по формуле 1)

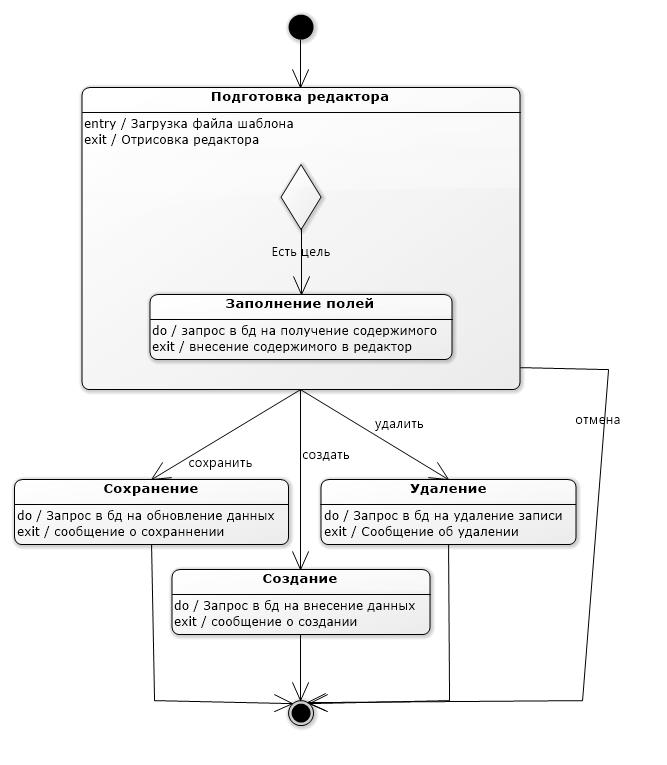


Рисунок 16 – Диаграмма состояний для класса Editor и всех его потомков

Все потомки этого класса работают по тому же типу, изменяется только тип данных, над которыми они работают

**Расчет оценки диаграммы**

Sdiagr = (по формуле 1)

## **4.5 Диаграммы видов деятельности**

Диаграмма видов деятельности - диаграмма, на которой показано разложение некоторой деятельности на ее составные части. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов - вложенных видов деятельности и отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла ко входам другого.

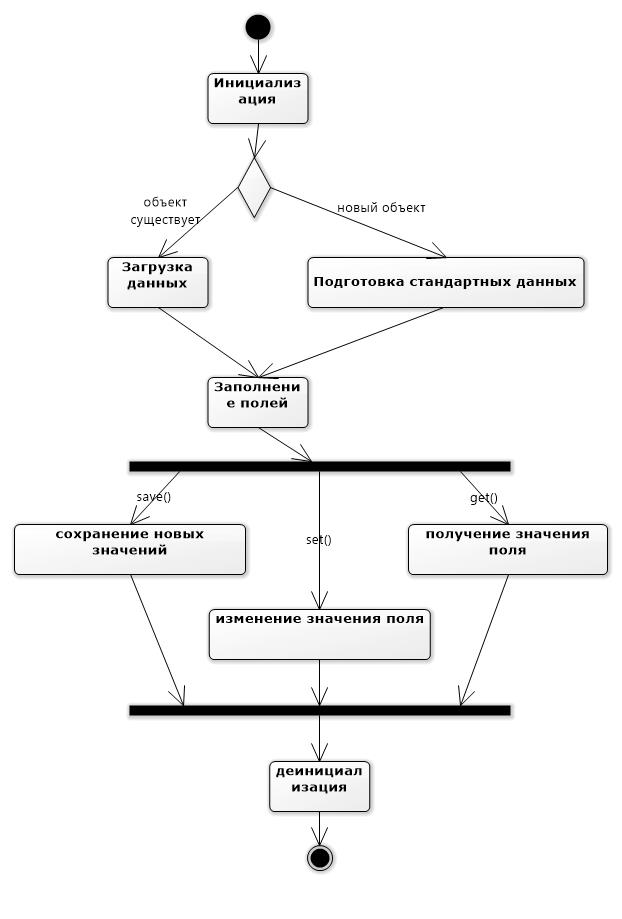
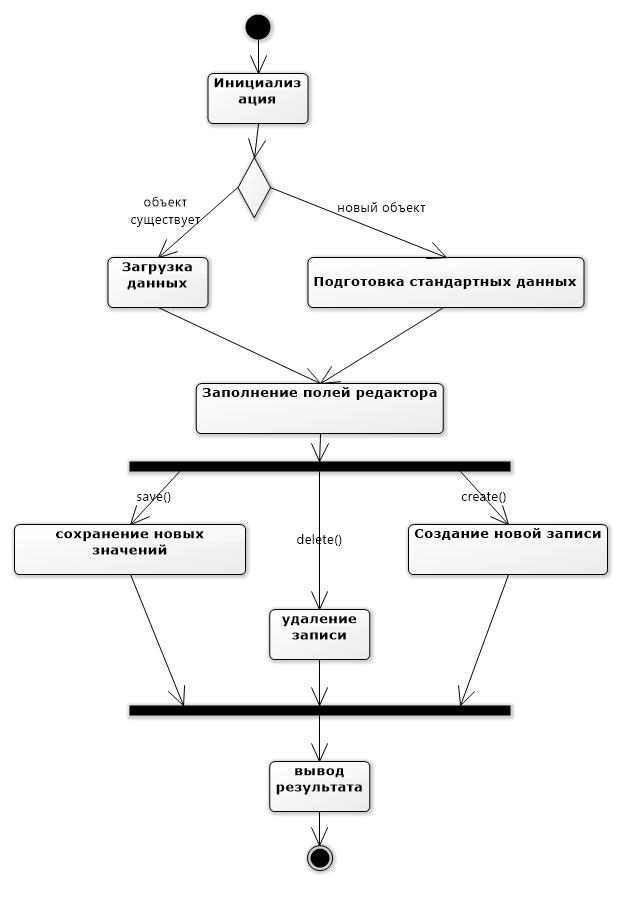


Рисунок 17 – Диаграмма видов деятельности класса “BaseElement”

**Расчет оценки диаграммы**

Sdiagr = 3.77 (по формуле 1)

Рисунок 18 – Диаграмма видов деятельности “Шлюз для игровых серверов” при обращении за данными о покупках



**Расчет оценки диаграммы**

Sdiagr = 3.77 (по формуле 1)

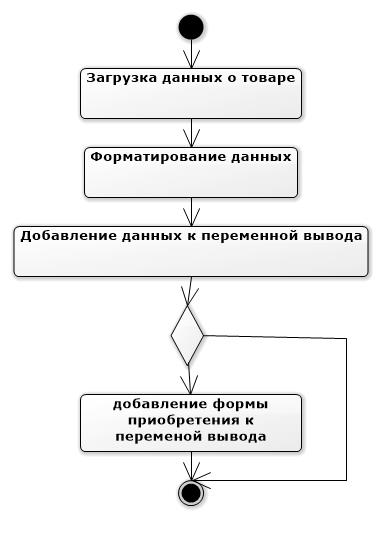


Рисунок 19 – Диаграмма видов деятельности для класса “ItemFullPrinter”

**Расчет оценки диаграммы**

Sdiagr = = 2.97 (по формуле 1)

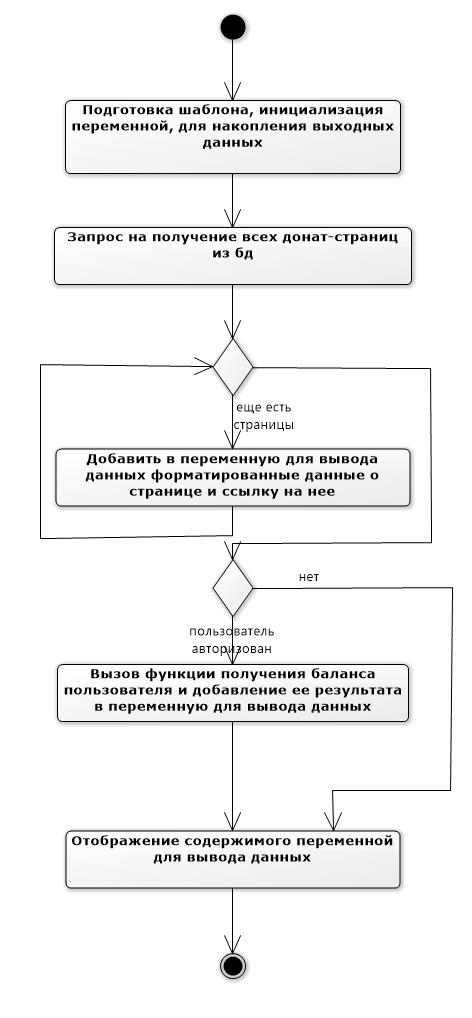


Рисунок 20 – Диаграмма видов деятельности класса “PageListPrinter”

**Расчет оценки диаграммы**

Sdiagr = = 3.08 (по формуле 1)

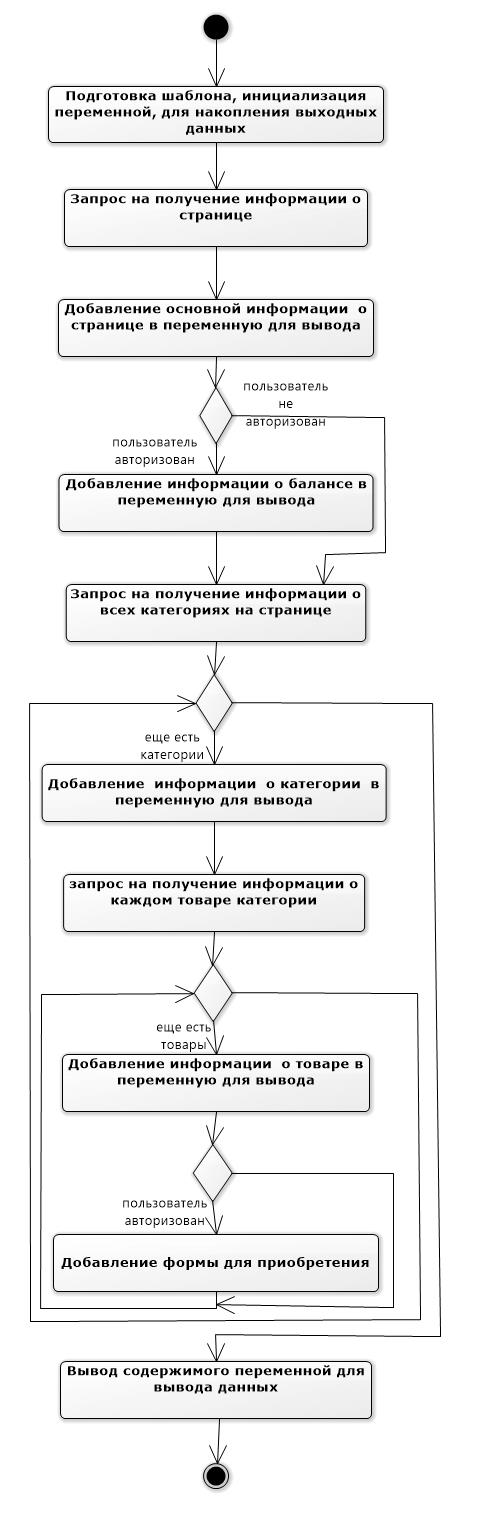


Рисунок 21 – диаграмма видов деятельности класса “PagePrinter”

**Расчет оценки диаграммы**

Sdiagr = 4.07 (по формуле 1)

## **4.6 Диаграмма размещения** Диаграмма развертывания (Диаграмма размещения) предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. Диаграмма развертывания отражает физические взаимосвязи между программными и аппаратными компонентами разрабатываемой системы. Каждый узел на диаграмме развертывания представляет собой некоторый тип вычислительного устройства - в большинстве случаев самостоятельную часть аппаратуры.

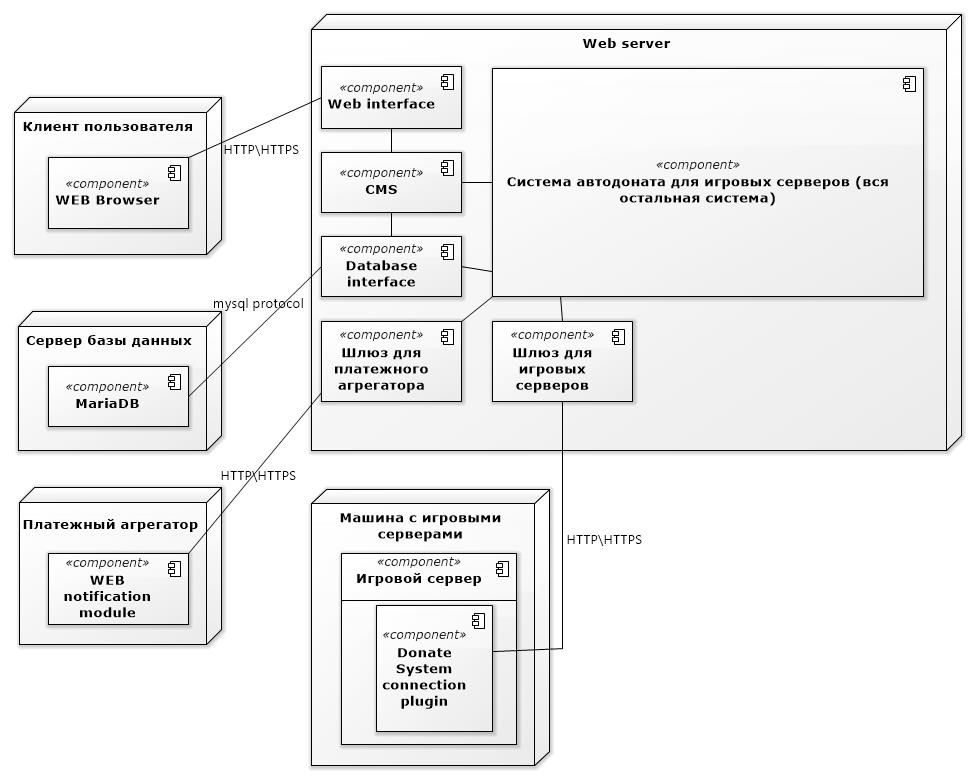


Рисунок 22 – диаграмма размещения

Данная диаграмма позволяет понять, на какие физические компоненты делится система, как ее компоненты расположены относительно друг друга и как они взаимодействуют.

**Расчет оценки диаграммы**

Sdiagr = = 3.68 (по формуле 1)

## **4.7 Диаграмма пакетов**

Диаграмма пакетов - структурная диаграмма, основным содержанием которой являются пакеты и отношения между ними. Диаграммы пакетов служат, в первую очередь, для организации элементов в группы по какому-либо признаку с целью упрощения структуры и организации работы с моделью системы. Строго говоря, пакеты и зависимости являются элементами диаграммы классов, т. е. диаграмма пакетов - это всего лишь форма диаграммы классов. Однако на практике причины построения таких диаграмм различны.

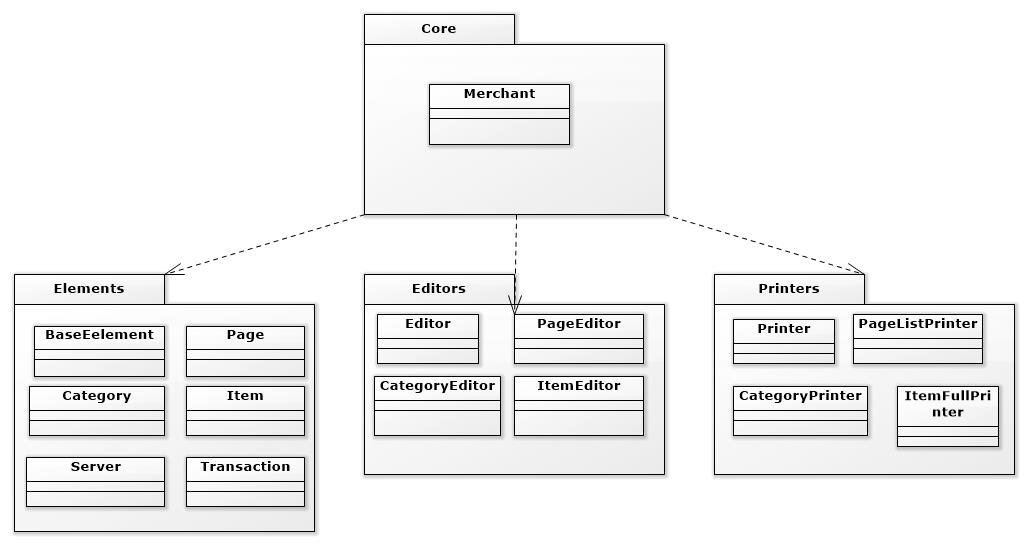


Рисунок 23 – диаграмма пакетов

**Расчет оценки диаграммы**

Sdiagr = (по формуле 1)

**ЗАКЛЮЧЕНИЕ**

В данной курсовой работе спроектирована структура программного обеспечения информационной системы «Система автодоната для игровых серверов» в нотации UML 2.0 с использованием Сase-средства Sofrware Ideas Modeler. Система описана практически со всех возможных точек зрения, рассмотрены разные аспекты поведения системы, диаграммы сравнительно просты для чтения, методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках.

Данная работа дает возможность организовать качественное функционирование описанной информационной системы, позволяет частично автоматизировать процесс создания программной реализации системы, что сэкономит и время, и денежные средства, необходимые для разработки.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Иванова Г.С. Технология программирования: учебник/Г.С. Иванова – М.:КНОРУС, 2011.-336с.
2. Леоненков А.В. Самоучитель UML: монография/ А. Леоненков. - СПб. и др.: BHV - Санкт-Петербург, 2014 - 298 с.
3. Буч Г., Якобсон А., Рамбо Дж. Язык UML. Руководство пользователя / Пер. с англ. — ДМК Пресс , 2014 г.
4. Software Ideas Modeler - Agile CASE tool for software design & analysis [Электронный ресурс]  
    <https://www.softwareideas.net/>
5. PHP: Hypertext preprocessor [Электронный ресурс]  
    <http://php.net/>

**ПРИЛОЖЕНИЕ**

П.1 Код на языке PHP, сгенерированных из диаграммы классов

<?php

//

abstract class BaseElement

{

//

public /\*String\*/ $primaryKey;

//

public /\*String\*/ $table;

//

public function load ()

{

}

//

public function save ()

{

}

//

public abstract function get (/\*object\*/ $param);

//

public abstract function set (/\*object\*/ $param);

}

?>

<?php

//

class Page extends BaseElement

{

//

public /\*Integer\*/ $id;

//

public /\*String\*/ $name;

}

?>

<?php

//

class Category extends BaseElement

{

//

public /\*object\*/ $id;

//

public /\*object\*/ $name;

}

?>

<?php

//

class Item extends BaseElement

{

//

public /\*Integer\*/ $id;

//

public /\*String\*/ $name;

//

public /\*String\*/ $content;

//

public /\*Integer\*/ $cost;

//

public /\*String\*/ $server\_commands;

//

public /\*DateTime\*/ $date;

}

?>

<?php

//

class Server extends BaseElement

{

//

public /\*Integer\*/ $id;

//

public /\*String\*/ $name;

}

?>

<?php

//

class transaction extends BaseElement

{

//

public /\*Integer\*/ $id;

//

public /\*String\*/ $player;

//

public /\*Boolean\*/ $done;

//

public /\*DateTime\*/ $date;

/\*Server\*/ $1;

}

?>

<?php

//

class PagePrinter implements Printer

{

//

public /\*object\*/ $page\_id;

//

private /\*object\*/ $template;

}

?>

<?php

//

class PageListPrinter implements Printer

{

//

private /\*object\*/ $template;

/\*object\*/ $\*;

}

?>

<?php

//

class ItemFullPrinter implements Printer

{

//

public /\*Integer\*/ $pack\_id;

//

public /\*Boolean\*/ $can\_buy;

//

private /\*object\*/ $template;

}

?>

<?php

//

class Merchant

{

//

public static function buy (/\*Item\*/ $item, /\*String\*/ $player, /\*Server\*/ $server)

{

}

}

?>

<?php

//

class Editor implements Printer

{

//

protected /\*BaseElement\*/ $target;

//

public function save ()

{

}

//

public function delete ()

{

}

//

public function create ()

{

}

}

?>

<?php

class PageEditor extends Editor

{

}

?>

<?php

class CategoryEditor extends Editor

{

/\*Category\*/ $1;

}

?>

<?php

class ItemEditor extends Editor

{

}

?>

<?php

//

interface Printer

{

//

public function print ();

}

?>