

Реферат

Пояснительная записка 72 страниц, 50 рисунков, 24 таблицы, 20 литературный источник, 3 приложений.

ВЕБ-ПРИЛОЖЕНИЕ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, JAVA, MAVEN, API, JSP, POSTGRESQL, HIBERNATE, SQL, TOMCAT, JWT, INTELLIJ IDEA, GIT, АУКЦИОН

Целью проекта является разработка веб-приложения для проведения аукционных торгов в сети интернет.

Пояснительная записка состоит из введения, шести разделов и заключения.

Во введении предоставлена краткая информация о появлении аукционных торгов и поставлены цели дипломного проекта.

В первом разделе дипломного проекта приведён обзор аналогов по теме дипломного проекта.

Во втором разделе приведено проектирование веб-приложения, диаграмма использования и описание спроектированной базы данных.

В третьем разделе приведено обоснование технических приемов программирования и реализация веб-приложения.

В четвертом разделе приведено тестирование веб-приложения.

В пятом разделе приведено руководство пользователя.

В шестом разделе приводится расчет экономических показателей.

В заключении приведены результаты проделанной работы.

| | | | | | | | | | | |
|-------------|------|---------------|---------|------|--------------|--|--|---------------------|------|--------|
| | | | | | ДП 00.00. ПЗ | | | | | |
| | | | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | Реферат | | | Лит. | Лист | Листов |
| Разраб. | | Жигало В. Ю. | | | | | | У | 1 | 1 |
| Провер. | | Наркевич А.С. | | | | | | БГТУ 74121013, 2018 | | |
| Консультант | | Наркевич А.С. | | | | | | | | |
| Н. Контр. | | Жиляк Н.А. | | | | | | | | |
| Утверд. | | Пацей Н. В. | | | | | | | | |

Содержание

| | |
|---|----|
| Введение | 6 |
| 1 Анализ предметной области | 8 |
| 1.1 Анализ существующих прототипов | 8 |
| 1.1.1 Веб-приложение «Ау» | 8 |
| 1.1.2 Веб-приложение «XLOT» | 9 |
| 1.1.3 Веб-приложение «Белаукцион» | 10 |
| 1.2 Основные требования к веб-приложению для проведения торгов | 11 |
| Выводы по разделу 1 | 11 |
| 2 Проектирование веб-приложения | 12 |
| 2.1 Требования к разрабатываемому проекту | 13 |
| 2.2 Проектирование | 13 |
| Выводы по разделу 2 | 18 |
| 3 Программная реализация | 19 |
| 3.1 Клиент-серверная архитектура проекта | 21 |
| 3.2 Реализация архитектуры проекта (описание пакетов и классов) | 22 |
| 3.3 Реализация соединения и взаимодействия с базой данных | 26 |
| 3.4 Реализация отправки писем, подключение к почтовому сервису | 30 |
| 3.5 Реализация создания PDF отчета | 31 |
| 3.6 Реализация создания Excel отчета | 31 |
| 3.7 Реализация механизма проведения торгов | 32 |
| 3.8 Реализация API | 35 |
| 3.9 Реализация шифрования (хэширования) данных | 38 |
| 3.10 Реализации модуля формирования статистики в виде графиков | 38 |
| 3.11 Реализация работы с cookies | 40 |
| Вывод по разделу 3 | 41 |
| 4 Тестирование веб-приложения | 42 |
| 4.1 Модульное тестирование | 44 |
| 4.2 Тестирование кода при сборке артефакта | 48 |
| Выводы по разделу 4 | 50 |
| 5 Руководство пользователя системы | 51 |
| 5.1 Гость | 52 |
| 5.2 Пользователь | 52 |
| 5.3 Администратор | 55 |
| Вывод по разделу 5 | 58 |
| 6.1 Общая характеристика разрабатываемого веб-приложения | 59 |
| 6.2 Исходные данные | 59 |
| 6.3.1 Объем веб-приложения | 61 |
| 6.3.2 Основная заработная плата | 62 |
| 6.3.3 Дополнительная заработная плата | 62 |

| | | | | | | | |
|-------------|-------------|-----------------|----------------|-------------|----------------------------|-------------|---------------|
| | | | | | <i>ДП 00.00. ПЗ</i> | | |
| | | | | | | | |
| <i>Изм.</i> | <i>Лист</i> | <i>№ докум.</i> | <i>Подпись</i> | <i>Дата</i> | <i>Содержание</i> | | |
| Разраб. | | Жигало В.Ю. | | | | | |
| Провер. | | Наркевич А.С. | | | | | |
| Консультант | | Наркевич А.С. | | | | | |
| Н. Контр. | | Жилияк Н.А. | | | | | |
| Утверд. | | Пацей Н.В. | | | <i>БГТУ 74121013, 2018</i> | | |
| | | | | | | | |
| | | | | | | <i>Лит.</i> | <i>Лист</i> |
| | | | | | | <i>У</i> | <i>1</i> |
| | | | | | | | <i>Листов</i> |
| | | | | | | | <i>2</i> |

| | |
|--|----|
| 6.3.4 Отчисления в Фонд социальной защиты населения и Белгосстрах | 62 |
| 6.3.5 Расходы на материалы..... | 63 |
| 6.3.6 Расходы на оплату машинного времени..... | 63 |
| 6.3.7 Прочие прямые затраты | 64 |
| 6.3.8 Накладные расходы | 64 |
| 6.3.9 Сумма расходов на разработку веб-приложения..... | 64 |
| 6.3.10 Расходы на сопровождение и адаптацию | 65 |
| 6.3.11 Полная себестоимость | 65 |
| 6.3.12 Определение цены, оценка эффективности | 66 |
| Вывод по разделу 6 | 66 |
| ЗАКЛЮЧЕНИЕ | 68 |
| ПРИЛОЖЕНИЕ А. Листинг исходного кода генерации отчета истории ставок на лот PDF..... | 71 |
| ПРИЛОЖЕНИЕ Б. Логическая схема базы данных..... | 73 |
| ПРИЛОЖЕНИЕ В. Блок-схема алгоритма регистрации и/или авторизации пользователя | 74 |

ВВЕДЕНИЕ

Во все времена торговля имела важное значение в жизни человека, а также во внешней торговле между странами.

Аукционная форма торгов применялась еще древними римлянами. После падения Римской империи данная форма торгов была забыта и возродилась только в середине XVII-го столетия, в Голландии. Голландская схема аукционных торгов отличалась от современной схемы торгов в том, что в Голландской схеме торги велись на понижение цены, а не повышение. В настоящее время, Голландская схема тогов уже не используется.

История говорит о том, что первый аукционный дом в Европе был основан в 1674 году. Также значимым моментов в истории можно выделить аукцион, открытый в 1707 году в Вене. К середине XVIII века, вся европа была заполнена аукционными домами. Самыми знаменитыми аукционными домами считаются Лондонские аукционные дома – Sothebys и Christies.

Организация аукционной торговли имеет отпределенные особенности, как и любой вид торгов. Аукционные дома занимаются организацией аукционов, на которых продаются как свои товары, так и коммисионные товары. При организации аукциона, организатор берет на себя все функции по подготовке и проведению аукционов. Единица товара на аукционе называется лот.

Техника проведения аукционных торгов по различным товарам имеет некоторые особенности, однако порядок проведения аукциона остается аналогичен. В аукционных торгах можно выделить 4 стадии: подготовка, осмотр лота, торги, оформление сделки. В качестве участников аукционных торгов могут выступать как физические, так и юридические лица. Идея аукциона заключается в том, чтобы создать конкуренцию и, следовательно, добиться роста цены за лот.

С появлением интернет сетей, появились интернет-аукционы, но это совсем не означает, что аукционные дома изжили себя. Интернет-аукционы – занимают малую часть от всех торгов, проводимых в инетрнете, число интернет-магазинов значительно превышает число интернет-аукционов. Из преимуществ интернет-аукциона можно выделить то, что аукцион проводится удаленно, что позволяет участникам находится в любой точке мира и участвовать в торгах. Момент окончания аукционных торгов в интеренете, заранее известен в отличии от традиционных торгов, проводимых аукционными домами. В традиционных аукционах борьба за лот идет до момента пока ставки повышаются, а в интернет-аукционе это может быть фиксированный промежуток времени.

Актуальность дипломного проекта связана с тем, что в настоящее время площадок для проведения аукционных торгов в инеретнете малое количество и со многими возможно конкурировать.

| | | | | | | | | | |
|-------------|------|---------------|---------|------|--------------|--|---------------------|------|--------|
| | | | | | ДП 00.00. ПЗ | | | | |
| | | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | | | | | |
| Разраб. | | Жигало В.Ю. | | | Введение | | Лит. | Лист | Листов |
| Провер. | | Наркевич А.С. | | | | | У | 1 | 2 |
| Консультант | | Наркевич А.С. | | | | | БГТУ 74121013, 2018 | | |
| Н. Контр. | | Жиляк Н.А. | | | | | | | |
| Утверд. | | Пацей Н.В. | | | | | | | |

Целью данного дипломного проекта является разработка веб-приложения «Аукцион».

В связи с поставленной целью необходимо разработать:

- серверную часть приложения;
- клиентскую часть приложения;
- архитектуру базы данных;
- авторизацию пользователя;
- регистрацию пользователя;
- поиск;
- добавление лота на торги;
- изменение лота;
- удаление лота;
- механизм ставок;
- генерация отчета в файл;
- профиль пользователя;
- администраторскую часть веб-приложения;
- управление лотами и пользователями;
- отображение статистики на графиках.

1 Анализ предметной области

Аукцион – вид продажи товара с публичных торгов с помощью посреднических организаций, располагающих специальным оборудованием, помещением и способствующих образованию рынков, где торговля ведется методом открытых торгов.

Организаторами аукциона могут выступать крупные торговые компании, ассоциации продавцов или специализированные аукционные брокерские фирмы. Организация и техника проведения аукциона на прямую зависит от товара и его характеристик, но сам порядок проведения аукциона одинаков. Выступающее в качестве организаторов торгов как правило приобретают продукцию, выставляемую на аукционах или, продают на аукционах товар по поручению заказчиков за комиссионные вознаграждения [1].

Главной стадией аукциона является торг, который проводит аукционист и его ассистенты. Выделяют два вида проведения торгов:

– гласный. При гласном способе проведения аукцион, аукционист объявляет номер очередного лота, называет начальную цену и спрашивает, кто готов заплатить больше установленной цены за этот лот. Покупатели повышают цену каждый раз на величину не ниже минимальной шага цены, который оговаривается заранее. Если очередного повышения цены не предлагается, аукционист после троекратного повторения вопроса – ударяет молотком, подтверждая, что данный лот продан последнему покупателю, предложившему наивысшую цену;

– негласный. При негласном способе проведения аукциона покупатели подают аукционисту условный знак, чаще всего это поднятия таблички с номером покупателя, о согласии поднять цену. Надбавка к цене стандартна и оговорена в правилах проведения торгов. Аукционист каждый раз объявляет новую цену, не называя покупателя.

1.1 Анализ существующих прототипов

1.1.1 Веб-приложение «Ау»

Одним из аналогов данного веб-приложения является веб-сервис «Ау.by» (<http://au.by>), главная страница представлена на рисунке 1.1. Ау.by это крупнейшая в Беларуси онлайн площадка для продажи и покупки товаров. Аукционы со ставками или продажа по фиксированной цене [2].

Сайт предоставляет пользователю возможность разместить лот на торги либо без проведения торгов (по принципу интернет-магазина). При размещении лота на торги пользователь указывает продолжительность торгов от 3 до 30 дней.

| | | | | | | | |
|--------------------|-------------|----------------------|----------------|-------------|---|-------------|---------------|
| | | | | | <i>ДП 01.00. ПЗ</i> | | |
| | | | | | | | |
| <i>Изм.</i> | <i>Лист</i> | <i>№ докум.</i> | <i>Подпись</i> | <i>Дата</i> | | | |
| <i>Разраб.</i> | | <i>Жигало В.Ю.</i> | | | <i>Аналитический обзор предметной области</i> | | |
| <i>Провер.</i> | | <i>Наркевич А.С.</i> | | | | | |
| <i>Консультант</i> | | <i>Наркевич А.С.</i> | | | | | |
| <i>Н. Контр.</i> | | <i>Жуляк Н.А.</i> | | | | | |
| <i>Утверд.</i> | | <i>Пацей Н.В.</i> | | | | | |
| | | | | | <i>Лит.</i> | <i>Лист</i> | <i>Листов</i> |
| | | | | | У | 1 | 4 |
| | | | | | <i>БГТУ 74121013, 2018</i> | | |

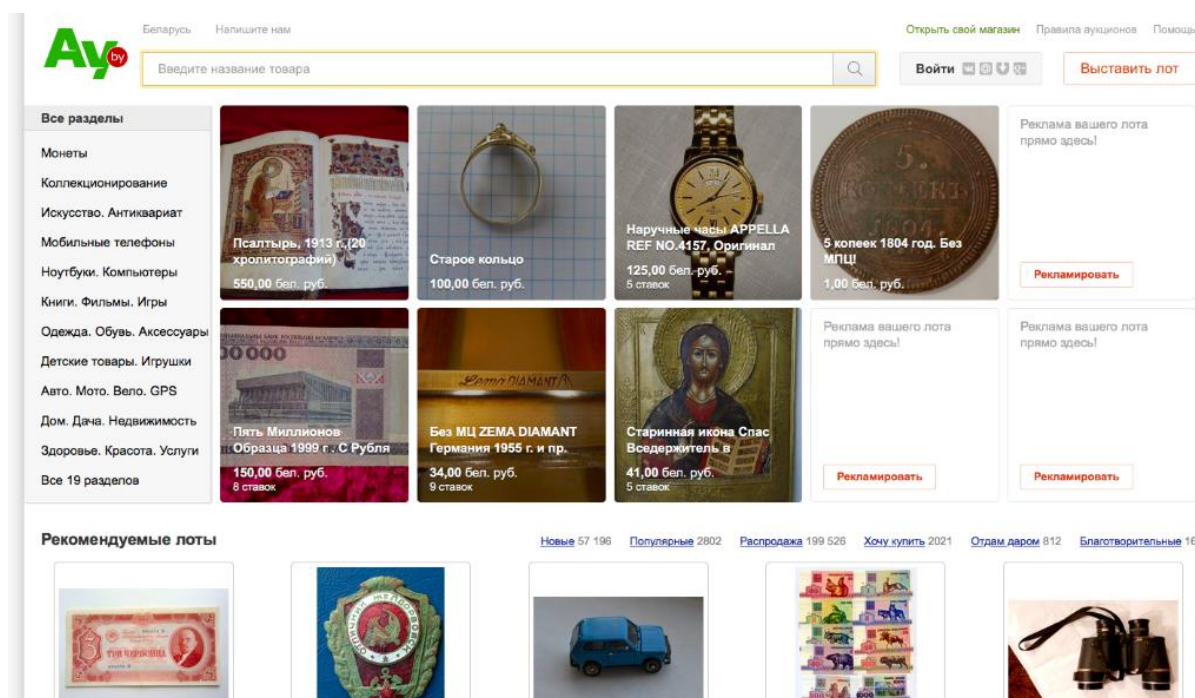


Рисунок 1.1 — Скриншот главной страницы сайта Ay.by

Ay.by функционирует с 2004 года и является популярной площадкой для продажи коллекционных товаров. API для сторонних разработчиков, сайт Ay.by не предоставляет.

Из недостатков приведенного выше сервиса, можно выделить фиксированное время торгов, разрабатываемое программное средство «Аукцион» позволит проводить торги в режиме реального времени, как это было, если бы торги проводились в аукционном доме. А именно, после ставки у пользователей будет отведено время для повышения ставки, и только после того, как ставок больше нет, лот считается проданным.

Из преимуществ можно выделить:

- большую клиентскую базу;
- огромный выбор лотов;
- возможность продажи товара как в интернет-магазине (без проведения торгов);
- возможность продажи товара на торгах.

1.1.2 Веб-приложение «XLOT»

XLOT – это ресурс реализующий конфискованные автомобили через сайт продажи ДилайнТоргСервис в Бресте, а также имущество, арестованное судебным исполнителем через аукцион. Скриншот главной страницы веб-приложения XLOT перставлен на рисунке 1.2 [3].

Из недостатков можно выделить то, что аукцион действует только в городе Бресте, что лишает жителей других городов полноценно участвовать в торгах, а также то, что веб-приложение сосредоточено только на продаже автомобилей, что сильно сужает число пользователей.

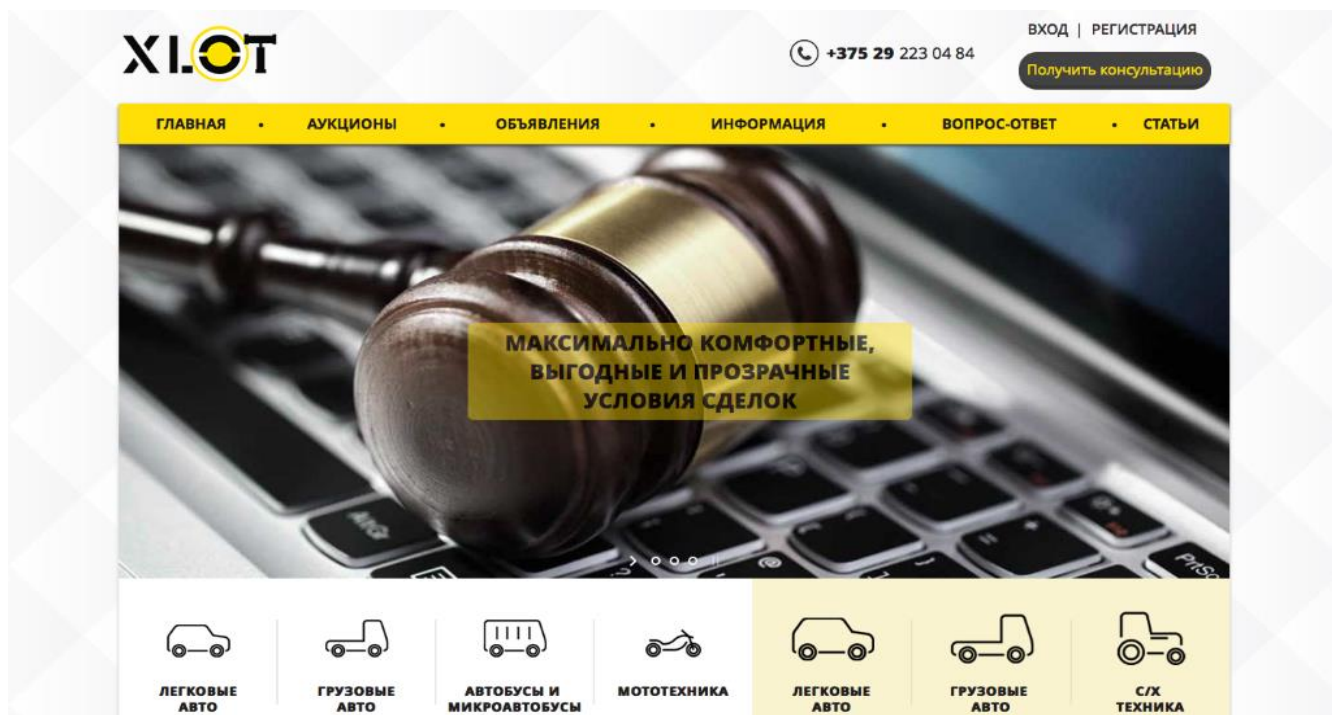


Рисунок 1.2 — Скриншот главной страницы сайта xlot.by

Из преимуществ можно выделить приятный интерфейс, возможность обратной связи с пользователями.

1.1.3 Веб-приложение «Белаукцион»

Белаукцион – аналог разрабатываемого веб-приложения. Главная страница веб-приложения представлена на рисунке 1.3 [4].

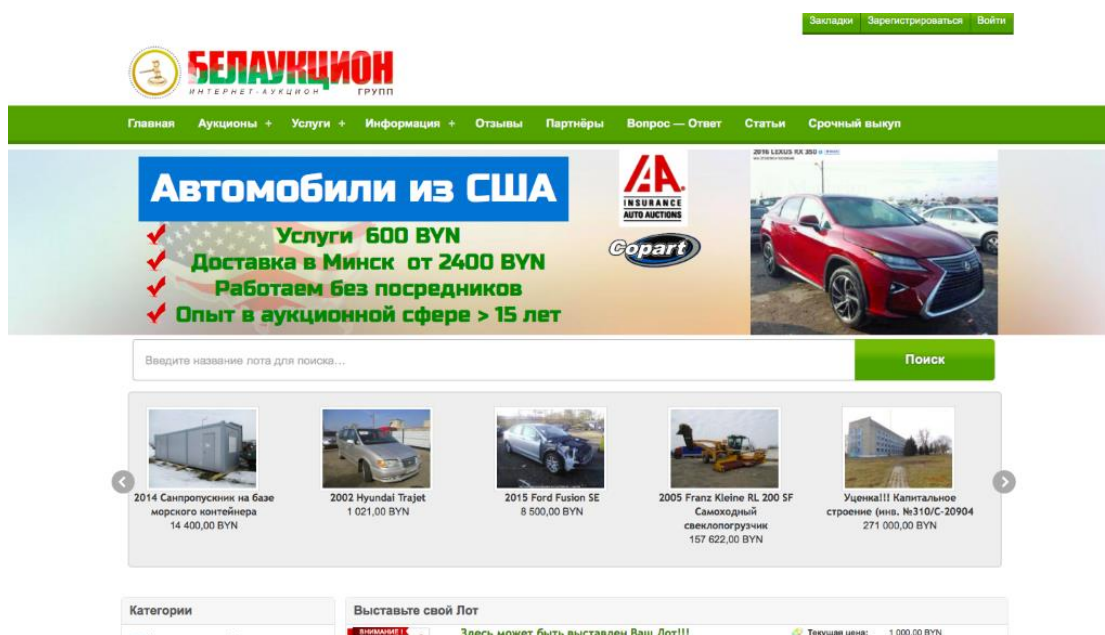


Рисунок 1.3 — Главная страница сайта belauktion.by

Данная площадка занимается реализацией через аукцион:

- аварийных автомобилей;
- автомобилей с пробегом;
- строительной и специальной техники;
- автопарков организаций, залогового имущества;
- имущества предприятий банкротов.

1.2 Основные требования к веб-приложению для проведения торгов

Как правило, в число основных функций, которые должны присутствовать в веб-приложении для проведения торгов (аукционные торги), следующие:

- добавление лотов;
- проведение торгов;
- уведомление продавца и покупателя о результате тооргов;
- статистика для администратора.

Выводы по разделу 1

В разделе проведен обзор существующих аналогов для проведения торгов, определены их преимущества и недостатки.

Описаны основные требования, выдвигаемые к веб-приложениям на тему аукционных торгов.

2 Проектирование веб-приложения

Веб-приложение, разрабатываемое в рамках данного дипломного проекта, предназначено для проведения аукционных торгов в online-режиме. Данное веб-приложение позволит выставлять лоты на торги, участвовать в торгах, производить ставки на лоты, предоставит API для сторонних разработчиков. Для администратора веб-приложения предоставит возможность удобного администрирования и получения статистики использования веб-приложения, генерацию отчетов в файл pdf или excel.

Веб-приложение ориентированно на пользователей, желающих продать или приобрести товар на аукционе. Данное программное средство позволит практически моментально продать или приобрести товар. Выдели стадии разработки веб приложения:

- анализ требований к веб-приложению. На данном этапе будут сформулированы цели и задачи, которые надо решить в рамках дипломного проектирования;

- проектирование веб-приложения. На этапе проектирования производится проектирование веб-приложения, строятся UML диаграммы, обозначающие основные возможности применения веб-приложения. На данном этапе производится выбор платформы для реализации веб-приложения, определяется язык программирования, фреймворк, проектируется база данных, а также интерфейс;

- реализация веб-приложения. На данном этапе происходит реализация веб-приложения, разрабатываются прототипы отдельных модулей целевого продукта. Результатом данного этапа является рабочий прототип веб-приложения;

- тестирование веб-приложения. На данном этапе производится проверка соответствия требований к веб-приложению с его работой, проверяется корректность выполнения логики, производится поиск ошибок. Для тестирования применяется ручное тестирование, когда тестирующий воспроизводит действия пользователя и дает заключение о соответствии веб-приложения заявленным требованиям. Также производятся следующие виды тестов: модульное тестирование, стресс-тестирование, тестирование интерфейса пользователя. Результатом тестирования веб-приложения является устранение всех недостатков системы и заключение о ее качестве;

- внедрение и поддержка. Данный этап предусматривает: установку веб-приложения, обучение пользователей использованию, эксплуатация, поддержку веб-приложения.

| | | | | | | | | | |
|-------------|------|---------------|---------|------|-------------------------------|---------------------|------|--------|--|
| | | | | | ДП 02.00. ПЗ | | | | |
| | | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | | | | | |
| Разраб. | | Жигало В.Ю. | | | Проектирование веб-приложения | Лит. | Лист | Листов | |
| Провер. | | Наркевич А.С. | | | | У | 1 | 7 | |
| Консультант | | Наркевич А.С. | | | | БГТУ 74121013, 2018 | | | |
| Н. Контр. | | Жуляк Н.А. | | | | | | | |
| Утверд. | | Пацей Н.В. | | | | | | | |

2.1 Требования к разрабатываемому проекту

Основной целью данного дипломного проекта является создание веб-приложения, которое позволит проводить торги в режиме online. Веб-приложение должно быть построено на клиент-серверной архитектуре, с возможностью предоставить API для сторонних разработчиков.

Можно выделить следующие основные требования к программному продукту:

- программное средство должно иметь интуитивно понятный интерфейс;
- серверная часть должна иметь возможность расширения функционала, в случае новых требований, без ущерба реализованному функционалу;
- программное средство должно предоставлять API для сторонних разработчиков.

2.2 Проектирование

Проектирование веб-приложения в данном дипломном проекте производилось с помощью 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. Диаграмма использования приведена на рисунке 2.1 [5][6][7].

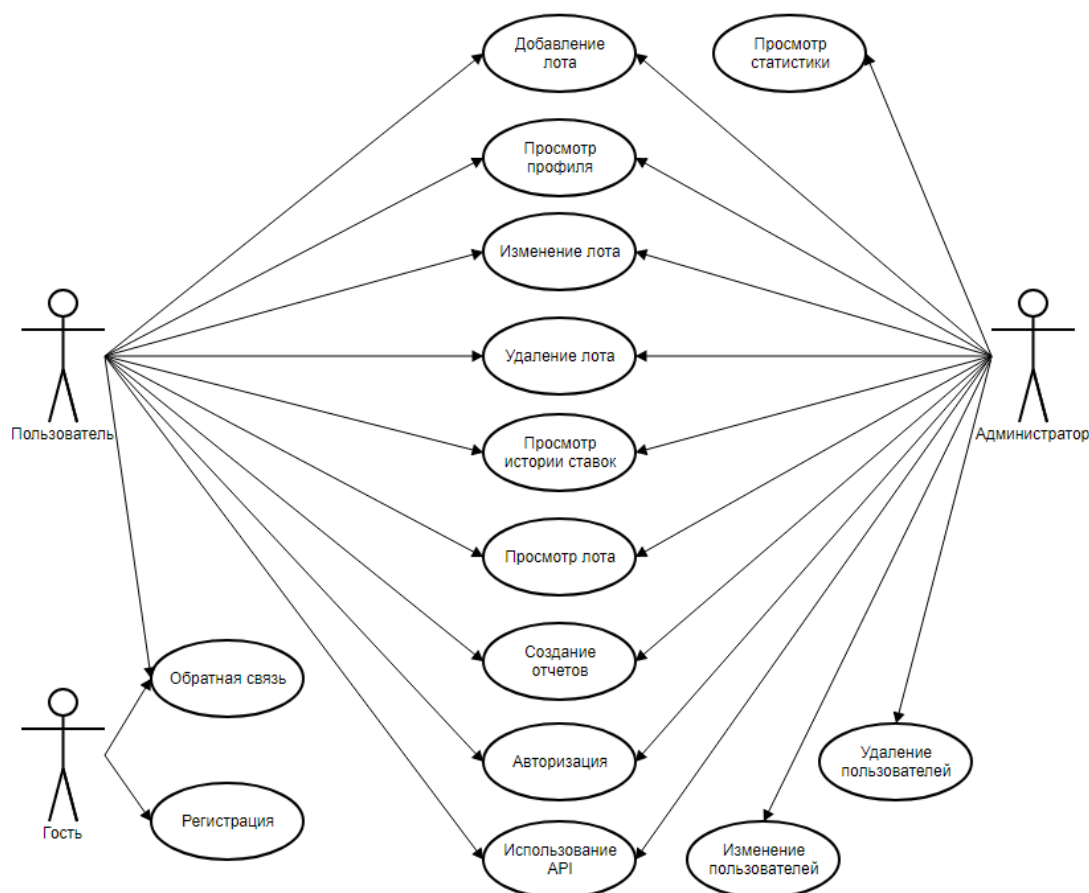


Рисунок 2.1 – Диаграмма использования веб-приложения

Для хранения информации о лотах, ставках, пользователях, отзывают необходимо использовать базу данных. Для этих целей была выбрана система управления базами данных PostgreSQL. Данная СУБД является свободно распространяемой и соответствует стандартам SQL и отличается полной поддержкой надежных транзакций: атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability) [9].

Выделим некоторые преимущества PostgreSQL над другими СУБД:

- является объектно-реляционной СУБД;
- поддерживает многомерные массивы. Это значит, что при определении типа столбца оно может быть массивом. Для создания столбца, хранящего массив необходимо при создании таблицы добавить “[]” к типу данных (например, title_column text []);
- позволяет хранить геометрические данные (тип столбца “path”);
- поддержка json. Это может быть удобно, когда база данных находится на этапе проектирования и до конца не известно какой тип данных будет хранить столбец. Стоит отметить, что тип данных JSON обеспечивает проверку корректности JSON;
- позволяет создать новый тип объекта;
- частичные индексы. Это удобно если разработчику требуется проиндексировать отдельное подмножество таблицы;
- функциональные индексы. Функциональные индексы позволяют предварительно вычислить столбец для индексирования;
- GIST (Generalized Search Tree) Обобщенное Дерево Поиска позволяет объединить B-деревья, R-деревья и пользовательские индексы для создания индекса с расширенными возможностями запросов;
- GIN (Generalized Inverted Index) Обобщенный Инвертированный Индекс позволяет проиндексировать составные типы данных;
- BRIN (Block Range Index) позволяет разделить большие таблицы на диапазоны основываясь на проиндексированный столбец;
- Расширенный функционал виртуальных таблиц;
- CTE (Common Table Expressions) поддержка запросов с выражением WITH;
- Позволяет использовать материализованные представления;
- Объединение запросов. PostgreSQL поддерживает условия INTERSECT и EXCEPT для взаимодействия между SELECT запросами, что расширяет возможности выборки из таблиц;
- оконные функции. Это функции, которые представляют собой агрегатные функции поверх некоторых строк выборки;
- латеральные вложенные запросы. Слово LATERAL может применяться ко вложенным запросам в условии FROM, с целью добавить перекрёстные ссылки между вложенным запросом и другими таблицами или виртуальными таблицами, которые были созданы ранее.

Вышеописанные преимущества обусловили мой выбор СУБД в сторону PostgreSQL. Как видно PostgreSQL имеет широкий функционал, является надежной и стабильной СУБД.

Рисунок связи таблиц базы данных представлен в приложении Б.

Как видно из рисунка представленного в приложении Б, база данных состоит из 7 таблиц, описание таблиц и столбцов приведены в таблицах 2.1 – 2.7.

Таблица `auth_info` предназначена для хранения информации для авторизации пользователей, описание полей представлено в таблице 2.1.

Таблица 2.1 – Описание столбцов таблицы «`auth_info`»

| Название | Описание |
|--------------------------|--|
| <code>id</code> | Уникальный номер записи в таблице, генерируется автоматически. Является первичным (<code>primary key</code>) ключом. |
| <code>login</code> | Поле содержит логин пользователя, используется для входа. |
| <code>password</code> | Поле содержит пароль пользователя, зашифрован с использованием хэш-функции <code>sha512</code> . |
| <code>email</code> | Поле содержит email пользователя. |
| <code>uuid</code> | Уникальный идентификатор пользователя. |
| <code>role</code> | Роль пользователя: администратор или пользователь. |
| <code>create_date</code> | Дата регистрации пользователя в веб-приложении. |
| <code>api_key</code> | Содержит уникальный ключ, используемый в <code>api</code> запросах. |
| <code>rate</code> | Поле содержит рейтинг пользователя. |

Таблица `personal_information` содержит персональную информацию о пользователях, описание столбцов представлено в таблице 2.2.

Таблица 2.2 – Описание столбцов таблицы «`personal_information`»

| Название | Описание |
|-------------------------|--|
| <code>id</code> | Уникальный номер записи в таблице, генерируется автоматически. Является первичным (<code>primary key</code>) ключом. |
| <code>uuid_user</code> | Уникальный идентификатор, однозначно связывающий пользователя из таблицы <code>auth_info</code> с данной таблицей. |
| <code>first_name</code> | Поле содержит имя пользователя. |
| <code>last_name</code> | Поле содержит фамилию пользователя. |
| <code>bday</code> | Поле содержит дату рождения пользователя. |
| <code>phone</code> | Поле содержит номер телефона пользователя. |

Таблица `address` содержит адреса пользователей, описание столбцов таблицы приведено в таблице 2.3.

Таблица 2.3 – Описание столбцов таблицы «`address`»

| Название | Описание |
|------------------------|--|
| <code>id</code> | Уникальный номер записи в таблице, генерируется автоматически. Является первичным (<code>primary key</code>) ключом. |
| <code>uuid_user</code> | Уникальный идентификатор, однозначно связывающий пользователя из таблицы <code>auth_info</code> с данной таблицей. |
| <code>country</code> | Поле содержит информацию о стране проживания, указанную пользователем. |
| <code>city</code> | Поле содержит информацию о городе, указанную пользователем. |

Продолжение таблицы 2.3

| | |
|--------|--|
| street | Поле содержит информацию об улице, пользователя. |
| house | Поле содержит информацию о номере дома, указанную пользователем. |
| zip | Поле содержит информацию об индексе, указанную пользователем. |

Таблица feedback содержит отзывы о лотах и пользователях, описание столбцов приведено в таблице 2.4.

Таблица 2.4 – Описание столбцов таблицы «feedback»

| Название столбца | Описание |
|------------------|---|
| id | Уникальный номер записи в таблице, генерируется автоматически. Является первичным (primary key) ключом. |
| uuid | Уникальный идентификатор лота или пользователя, для которого оставлен отзыв. |
| id_user | Идентификатор пользователя, оставившего отзыв. |
| feedback_text | Поле содержит текст отзыва. |
| date | Дата, когда был оставлен отзыв. |

Таблица category содержит категории лотов, описание столбцов таблицы приведено в таблице 2.5.

Таблица 2.5 – Описание столбцов таблицы «category»

| Название столбца | Описание |
|------------------|---|
| id | Уникальный номер записи в таблице, генерируется автоматически. Является первичным (primary key) ключом. |
| name | Поле содержит название категории. |

Таблица bet содержит историю ставок на лот, описание столбцов в таблице приведено в таблице 2.6.

Таблица 2.6 – Описание столбцов таблицы «bet»

| Название столбца | Описание |
|------------------|---|
| id | Уникальный номер записи в таблице, генерируется автоматически. Является первичным (primary key) ключом. |
| uuid | Уникальный идентификатор лота, однозначно определяет лот из таблицы lot. |
| bult | Поле хранит JSON строку, содержащую историю ставок. |

Таблица lot содержит информацию о лотах, описание столбцов таблицы приведено в таблице 2.7.

Таблица 2.7 – Описание столбцов в таблице «lot»

| Название столбца | Описание |
|------------------|---|
| id | Уникальный номер записи в таблице, генерируется автоматически. Является первичным (primary key) ключом. |

Продолжение таблицы 2.7

| | |
|------------------|--|
| uuid | Уникальный идентификатор лота. Используется для идентификации лота и связан с таблицей bet. |
| uuid_user_seller | Уникальный идентификатор пользователя, добавивший лот. Связан с таблицей auth_info, поле в таблице auth_info – uuid. |
| name | Название лота |
| information | Информацию о лоте, описание лота. |
| cost | Стартовая цена за лот. |
| blitz_cost | Блиц-цена за лот. |
| step_cost | Минимальная сумма ставки за лот. |
| date_add | Дата добавления лота в базу. |
| date_start | Дата начала торгов. |
| date_end | Дата окончания торгов. |
| time_start | Время начала торгов. |
| time_end | Время окончания торгов. |
| uuid_user_client | Уникальный идентификатор пользователя, победивший в торгах. |
| id_category | Категория, к которой относится лот. |
| status | Статус лота, может иметь одно из четырех состояний: Доступен для ставок, в ожидании, продано, закрыт. |
| rate | Числовая величина, отображающая рейтинг лота. |
| image_name | Изображение лота. |

Пример таблицы bet из базы данных представлен на рисунке 2.2.

| | id | uuid | bulk |
|---|----|--------------------------------------|--|
| 1 | 0 | d83a7aa9-a099-46e1-94a9-af145ac54b8e | {"uuid_lot": "d83a7aa9-a099-46e1-94a9..."} |
| 2 | 16 | 5a132e1d-10c5-486c-886b-756fb4b3a1f8 | {"uuid_lot": "5a132e1d-10c5-486c-886b..."} |
| 3 | 17 | 53e1b135-8f6b-4af2-8bbd-544488438c09 | {"uuid_lot": "53e1b135-8f6b-4af2-8bbd..."} |
| 4 | 18 | c5372f48-478d-4eec-8df4-9ca71cadb529 | {"uuid_lot": "c5372f48-478d-4eec-8df4..."} |

Рисунок 2.2 – таблица для хранения информации о ставках

Для генерации уникального идентификатора использовалась Java библиотека UUID, данная библиотека входит в стандартный набор Java библиотек.

При добавлении лота в базу, сразу генерируется JSON-строка, содержащая некоторую информацию о лоте, пользователе, который создал лот, первый элемент в массиве ставок – указывает на пользователя, создавшего лот, и указавший стартовую цену.

Веб-приложение будет разработано в соответствии с клиент-серверной архитектурой. Архитектура проекта спроектирована так, что в случае требования расширить функционал, это сделать не составит труда.

Классы содержат методы, которые выполняют задачи, независимо друг от друга, что позволит использовать уже готовый метод, для реализации новых требований. Классы разделены на пакеты, четко отражающие суть содержащих классов. Классы содержат методы, наименование которых отражает суть метода.

Весь исходный код написан в соответствии с соглашением об написании Java кода. Константные переменные вынесены в утилитный класс – VariablesUtil, статические методы вынесены в класс – CommonUtil.

Выводы по разделу 2

Исходя из требований данного веб-приложения, можно сделать заключение о том, что хранение данных в приложении лучше всего организовать, используя СУБД PostgreSQL. Данная система организывает хранение данных в таком виде, который позволит эффективно реализовать работ с данными.

Веб-приложение будет разработано в соответствии с клиент-серверной архитектурой.

В разделе приведено описание таблиц базы данных, также приведено краткое описание столбцов.

3 Программная реализация

3.1 Выбор среды разработки и языка программирования

Для выбора среды разработки необходимо определиться с используемым языком программирования.

Для реализации веб-приложения исходя из требований, был выбран язык программирования Java. Java – это объектно-ориентированный язык программирования. Приложения, написанные на данном языке, транслируются в байт-код, что позволяет их запустить на любом устройстве с установленной Java-машиной.

Выделим преимущества выбранного языка программирования перед другими языками:

- автоматическое управление памятью;
- платформонезависимый;
- безопасным;
- многопоточный;
- интерпретированный;
- динамический;
- высокопроизводительный.

Для разработки веб-приложения была выбрана среда разработки от компании JetBrains – IntelliJ IDEA. IDE является платной, но для студентов предоставляется полная (Ultimate) версия в которой присутствуют все необходимые компоненты для разработки веб-приложений. Данная среда разработки доступна для Windows, macOS и Linux.

Выделим преимущества IntelliJ IDEA по сравнению с другими средами разработки:

- глубокое понимание кода – это означает, что IDE отображает список наиболее релевантных символов, применимых в данном контексте;
- встроенные инструменты – это означает, что IDE обеспечивает единый интерфейс взаимодействия с большинством систем контроля версий, а также возможность работы с базой данных непосредственно из IDE;
- поддержка языков программирования, в настоящий момент существует 19 языковых плагинов для IntelliJ IDEA.

Для создания веб-приложения используется технология JSP. JSP (JavaServer Pages), данная технология позволяет в кратчайшие сроки создавать веб-приложения. Одно из преимуществ JSP, это возможность создавать разметку с использованием Java кода, это означает что содержимое страницы разделяется на статический html код и динамический Java код [15] [16]. Синтаксис JSP страниц требует специальные теги, в которые будет заключен Java код. JSP страницы поддерживают 3 типа директив. Описание синтаксиса приведенное в таблице 3.1.

| | | | | | | | | |
|-------------|------|---------------|---------|------|--|------|---------------------|--------|
| | | | | | ДП 03.00. ПЗ | | | |
| | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | Программная реализация веб-приложения | Лит. | Лист | Листов |
| Разраб. | | Жигало В.Ю. | | | | У | 1 | 23 |
| Провер. | | Наркевич А.С. | | | | | | |
| Консультант | | Наркевич А.С. | | | | | | |
| Н. Контр. | | Жиляк Н.А. | | | | | | |
| Утверд. | | Пацей Н.В. | | | | | БГТУ 74121013, 2018 | |

Таблица 3.1 – Описание тегов JSP страницы

| Синтаксис | Краткое описание |
|---|--|
| <%= выражение %> | Выражение обрабатывается и будет направлено в стандартный поток вывода. Результат работы выражения имеет обязательный тип String. |
| <% код %> | В данные теги разработчик может поместить любой Java код. Например это объявления переменных, вызов методов. |
| <%! объявление переменной %> | Тег предназначен для объявления переменных, которые будут использованы в скриплетах JSP страницы. |
| Директивы | |
| <%@taglib uri="URI библиотеки тегов" prefix="Имя префикса" %> | Позволяет подключить библиотеку тегов к JSP странице, префикс ставит в соответствие библиотеку тегов, с помощью которого она будет использована на текущей странице. |
| <%@include file=" URI включаемой страницы" %> | Позволяет подключить файл к JSP странице. |
| <%@ page атрибуты %> | Позволяет определить свойства текущей JSP страницы. Свойства будут обработаны на этапе транслирования кода в байт-код. Директива поддерживает следующие атрибуты: <ul style="list-style-type: none"> – language; – autoFlush; – isThreadSafe; – extends; – import; – Session; – info; – errorPage; – isErrorPage; – Buffer; – contentType; – pageEncoding. |

Для обработки запросов используются сервлеты. Сервлет – это Java класс, наследующий класс HttpServlet и переопределяющий методы doGet, doPost и т.д. Взаимодействие сервлета с клиентом происходит посредством запрос-ответов. Жизненный цикл сервлета состоит из трех этапов.

- Если сервлет отсутствует в контейнере, он будет загружен механизмом class loader и контейнер вызовет метод `init()`, разработчик может переопределить этот метод, что позволит выполнить логику на этапе загрузки. Метод `init()` выполняется однажды.

- Обработка клиентского запроса. Каждый запрос обрабатывается в отдельном потоке. При поступлении запроса на сервлет вызывается метод `service()`, задача метода определить тип поступившего запроса (GET, POST, PUT, DELETE и etc). Если не переопределен метод для поступившего запроса, будет вызван метод родительского класса, который вернет ошибку инициатору запроса.

- Если контейнеру необходимо удалить сервлет, вызывается метод `destroy()`. Данный метод вызывается однажды, аналогично методу `init()`.

Для сборки проекта используется Apache Maven. Maven – это фреймворк, разработанный компанией Apache, предназначенный для автоматизации сборки проектов на основе файла на языке POM (Project Object Model). POM язык является подмножеством языка XML. Apache Maven позволяет собрать проект, создать jar/war файл, сгенерировать документацию и дистрибутив программы [13] [14].

Хотелось бы выделить преимущества Apache Maven:

- удобное управление зависимостями;
- независимость от операционной системы;
- поддержка сборки из командной строки (что позволяет записать строку в bat/sh скрипт);
- хорошая интеграция с популярными средами разработки.

Каждая зависимость, подключаемая к проекту, с использованием Maven, содержит следующие параметры:

- `groupId` – название организации, которая разработала плагин;
- `artifactId` – отображает название проекта;
- `version` – версия, которую требуется подключить.

Для развертывания веб-приложения выбран сервер Apache Tomcat. Tomcat – это контейнер сервлетов с открытым исходным кодом, разработанный компанией Apache. Сервер позволяет запускать веб-приложения, также содержит средства для самоконфигурирования [12].

Система контроля версий Git. GitHub – это сервис, в основе которого лежит Git. GitHub позволяет разработчикам бесплатно создавать публичные репозитории. Репозиторий – это хранилище исходного кода, публичный репозиторий доступен всем без исключения.

3.1 Клиент-серверная архитектура проекта

Дипломный проект реализован на клиент-серверной архитектуре. Клиент-серверная архитектура – это архитектура веб-приложений, которая позволяет разделить вычислительную нагрузку на сервера и клиентов. Инициатором запросов всегда является клиент, сервер ожидает запроса от клиента.

Сервер – это машина, обладающая вычислительной мощностью, для обработки запросов сразу от нескольких клиентов.

Клиент – программное обеспечение, позволяющее взаимодействовать с определенным сервером. Зачастую клиент и сервер может быть установлен на одной машине. Диаграмма взаимодействия клиента и сервера представлена на рисунке 3.1.

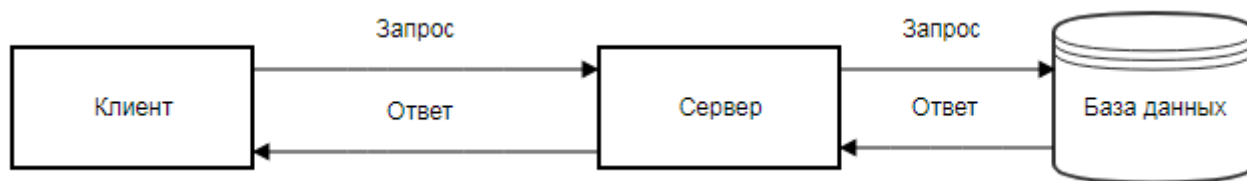


Рисунок 3.1 – Диаграмма, отражающая принцип клиент-серверной архитектуры

Из преимуществ клиент-серверной архитектуры можно выделить следующее:

- избежание дублирования кода на стороне клиента и на стороне сервера;
- безопасность. Все данные хранятся на сервере, который обладает большей защищенностью, по сравнению с клиентами;
- низкие требования вычислительной мощности к клиенту.

Недостатки клиент-серверной архитектуры:

- в случае если сервер по каким-то причинам перестанет работать, клиенты не смогут взаимодействовать с веб-приложением;
- сервер подразумевает высокую вычислительную мощность, из этого следует высокая стоимость серверов;
- поддержка данной архитектуры требует специалиста.

3.2 Реализация архитектуры проекта (описание пакетов и классов)

В языке программирования Java все классы с исходным кодом располагаются по пакетам. Пакеты позволяют логически объединить классы. Каждый класс имеет два названия:

- простое. Это имя класса, данное при его создании;
- полное. Это имя класса включающее имя пакета, в котором класс находится.

Структура пакетов точно отображает файловую систему. Все файлы с исходными кодами, входящие в один пакет, хранятся в одном каталоге файловой системы. Каждый пакет создает единое пространство имен. Это говорит о том, что все имена классов и интерфейсов в пакете должны быть уникальны. Структура пакетов в веб-приложении приведена на рисунке 3.2.

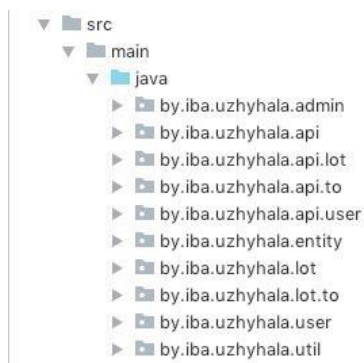


Рисунок 3.2 – Скриншот пакетов разрабатываемого веб-приложения

Описание пакетов и классов, которые там содержатся представлены в таблицах 3.2 – 3.11.

В пакете `by.iba.uzhyhala.admin` находятся классы, в которых реализована логика, относящаяся к администраторской части веб-приложения. Описание классов приведено в таблице 3.2.

Таблица 3.2 – Описание классов в пакете `by.iba.uzhyhala.admin`

| Название класса | Описание класса |
|------------------|--|
| FeedbackHandler | В классе реализованы методы для обработки отзывов: добавление, изменение, удаление. |
| RateHandler | В классе реализованы методы для изменения рейтинга пользователя и лот. |
| StatisticHandler | В классе реализованы методы для подготовки данных статистики в требуемом формате для графиков. |

В пакете `by.iba.uzhyhala.api.lot` находятся классы, в которых реализована логика, относящаяся к работе с лотами посредством API запросов. Описание классов приведено в таблице 3.3.

Таблица 3.3 – Описание классов в пакете `by.iba.uzhyhala.api.lot`

| | |
|--------------------------|---|
| LotAddAPI | В классе реализованы методы для добавления лота через API запрос. |
| LotBetHistoryDocumentAPI | В классе реализованы методы для получения документа с историей ставок через API запрос. Сервер вернет документ в виде закодированной строки Base64. |
| LotAddTOAPI | Класс является POJO, предназначен для промежуточного хранения разобранного тела запроса в формате JSON. |

В пакете `by.iba.uzhyhala.to` находятся утилитные POJO классы, которые используются в промежуточной обработке данных. Описание классов приведено в таблице 3.4.

Таблица 3.4 – Описание классов в пакете `by.iba.uzhyhala.to`

| | |
|-------------------|---|
| LotFullFieldTOAPI | Класс является POJO, предназначен для промежуточного хранения разобранного тела запроса в формате JSON. |
| LotTOAPI | Класс является POJO, предназначен для промежуточного хранения разобранного тела запроса в формате JSON. |
| UserRegTOAPI | Класс является POJO, предназначен для промежуточного хранения разобранного тела запроса в формате JSON. |

В пакете `by.iba.uzhyhala.api.user` находятся классы, которые используются для взаимодействия с пользователями посредством API запросов. Описание классов приведено в таблице 3.5.

Таблица 3.5 – Описание классов в пакете `by.iba.uzhyhala.api.user`

| | |
|-----------------|---|
| RegistrationAPI | В классе реализованы методы для регистрации пользователя в веб-приложении через API запрос. |
| UserApiKey | В классе реализованы методы для получения API ключа. |

В пакете `by.iba.uzhyhala.api` находятся универсальные классы, относящиеся к реализации API в разрабатываемом веб-приложении. Описание классов приведено в таблице 3.6.

Таблица 3.6 – Описание классов в пакете `by.iba.uzhyhala.api`

| | |
|------------------|---|
| DeleteHandlerAPI | В классе реализованы методы для удаления лота или пользователя через API. |
|------------------|---|

В пакете `by.iba.uzhyhala.entity` находятся утилитные POJO классы, которые используются в промежуточной обработке данных. Также классы из данного пакета отображают сущность таблиц базы данных. Описание классов приведено в таблице 3.7.

Таблица 3.7 – Описание классов в пакете `by.iba.uzhyhala.entity`

| | |
|---------------------------|---|
| AddressEntity | Класс является POJO, используется для отображения таблицы <code>address</code> из базы данных. |
| AuthInfoEntity | Класс является POJO, используется для отображения таблицы <code>auth_info</code> из базы данных. |
| BetEntity | Класс является POJO, используется для отображения таблицы <code>bet</code> из базы данных. |
| CategoryEntity | Класс является POJO, используется для отображения таблицы <code>category</code> из базы данных. |
| FeedbackEntity | Класс является POJO, используется для отображения таблицы <code>feedback</code> из базы данных. |
| LotEntity | Класс является POJO, используется для отображения таблицы <code>lot</code> из базы данных. |
| PersonalInformationEntity | Класс является POJO, используется для отображения таблицы <code>personal_information</code> из базы данных. |

В пакете `by.iba.uzhyhala.lot.to` находятся утилитные POJO классы, которые используются в промежуточной обработке данных. Описание классов приведено в таблице 3.8.

Таблица 3.8 – Описание классов в пакете by.iba.uzhyhala.lot.to

| | |
|--------------|---|
| BetBulkTO | Класс является POJO, предназначен для промежуточного хранения разобранного тела запроса в формате JSON. |
| BetHistoryTO | Класс является POJO, предназначен для промежуточного хранения разобранного тела запроса в формате JSON. |
| BetTO | Класс является POJO, предназначен для промежуточного хранения разобранного тела запроса в формате JSON. |

В пакете by.iba.uzhyhala.lot находятся классы, в которых реализована логика, относящаяся к работе с лотами и ставками на лот. Также пакет содержит класс реализующий логику генерации отчета об истории ставок. Описание классов приведено в таблице 3.9.

Таблица 3.9 – Описание классов в пакете by.iba.uzhyhala.lot

| | |
|-----------------|--|
| BetHandler | В классе реализованы методы для создания ставки на лот. Проверяется доступность ставки, проводится валидация. |
| DocumentHandler | В классе реализованы методы для генерации документа, содержащего историю ставок. |
| LotControl | В классе реализованы методы для подсчета времени торгов на лот. |
| LotHandler | В классе реализованы методы для добавления лота. Проводится валидация введенных данных. Реализован метод подготовки JSON строки, которая будет содержать историю ставок. |
| LotStatus | В классе реализованы методы для изменения статуса лота. |

В пакете by.iba.uzhyhala.user находятся классы, в которых реализована логика, относящаяся к работе с пользователями, а именно регистрация, авторизация, профиль, выход из веб-приложения. Описание классов приведено в таблице 3.10.

Таблица 3.10 – Описание классов в пакете by.iba.uzhyhala.user

| | |
|---------------|--|
| Authorization | В классе реализованы методы для авторизации пользователя в веб-приложении. Реализован метод создания куки. |
| Logout | В классе реализованы методы для выхода пользователя из веб-приложения. Реализован метод удаления куки. |
| Profile | В классе реализованы статические методы для вывода информации в профиле пользователя. |
| Registration | В классе реализованы методы для регистрации пользователя в веб-приложении. |

В пакете by.iba.uzhyhala.api.util находятся утилитные классы, в которых реализована логика методов, которые могут использоваться в различных частях веб-приложения. Методы являются универсальными. Описание классов приведено в таблице 3.11.

Таблица 3.11 – Описание классов в пакете `by.iba.uzhyhala.util`

| | |
|---------------|--|
| CommonUtil | Класс является «утилитным». В классе реализованы статические методы. Данные методы независимы от других методов и выполняют конкретную задачу. |
| CookieUtil | Класс является «утилитным». В классе реализованы методы для обработки cookie. |
| HibernateUtil | Класс является «утилитным». В классе реализован метод для взаимодействия с базой данных посредством Hibernate. |
| MailUtil | Класс является «утилитным». В классе реализованы методы для отправки электронных писем. Письма поддерживают формат html и plain text. |
| ReCaptchaUtil | Класс является «утилитным». В классе реализован метод для разбора проверки каптчи. |
| VariablesUtil | Класс является «утилитным». В классе определены константы. |

Из приведенных выше таблиц, мы видим, что архитектура проекта спроектирована так, что расширение функционала не вызовет затруднений. Название пакетов четко отражает суть классов, которые в нем содержатся.

3.3 Реализация соединения и взаимодействия с базой данных

Для работы с базой данных с использованием ORM-подхода, была выбрана библиотека Hibernate [10].

ORM (Object-Relational Mapping) – объектно-реляционное отображение означает, что разработчику удобнее работать с объектами, нежели чем с таблицами, в которых хранятся данные в реляционных базах данных. Для связи реляционных СУБД с объектной моделью приложения используются ORM-технологии. Так, для решения задач объектно-реляционного отображения в данном программном средстве используется библиотека Hibernate [11].

Hibernate является самым популярным ORM-решением для языка Java, которая не только отвечает за связь Java классов с таблицами базы данных, но и предоставляет возможности для автоматического построения запросов и извлечения данных из базы, что положительно сказывается на времени разработки.

Hibernate генерирует SQL запросы и освобождает разработчика от ручной обработки результирующего набора данных и конвертации объектов, сохраняя приложение портируемым во все SQL совместимые базы данных. Архитектура приложения, использующего приведена на рисунке 3.3.

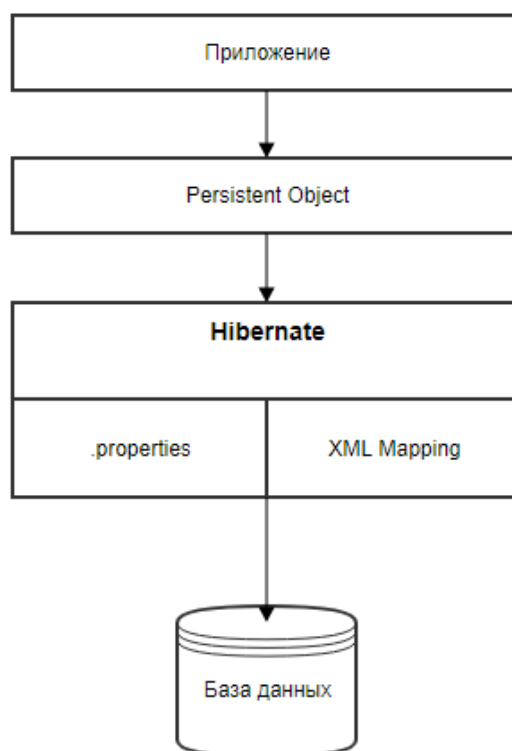


Рисунок 3.3 – Схема использования Hibernate в дипломном проекте

Для работы с базой данных посредством Hibernate необходимо выполнить следующие действия. Подключить Hibernate библиотеку к проекту, IntelliJ IDEA позволяет это сделать при создании проекта. Для обращений к базе данных через ORM – подход, необходимо создать вспомогательный класс, представлен на рисунке 3.4.

```

public class HibernateUtil {
    private static final Logger LOGGER = Logger.getLogger(HibernateUtil.class);
    private static final SessionFactory sessionFactory = buildSessionFactory();

    HibernateUtil() {
    }

    private static SessionFactory buildSessionFactory() {
        try {
            return new Configuration().configure(HIBERNATE_CONFIG).buildSessionFactory();
        } catch (RuntimeException ex) {
            LOGGER.error("Initial SessionFactory creation failed.\t" + ex.getLocalizedMessage());
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
  
```

Рисунок 3.4 – Скриншот исходного кода вспомогательного класса HibernateUtil

Создать файл конфигурации в формате xml. Описание свойств приводится в таблице 3.12.

Таблица 3.12 – Описание используемых свойств в файле конфигурации Hibernate

| Имя свойства | Описание |
|-----------------------------------|---|
| hibernate.connection.url | JDBC URL. |
| hibernate.connection.driver_class | Класс JDBC-драйвера. |
| hibernate.dialect | Подключает определенную платформозависимую функциональность, а именно поддержку SQL для конкретно указанной базы данных |
| hibernate.connection.username | Имя пользователя базы данных. |
| hibernate.connection.password | Пароль пользователя базы данных. |
| hibernate.show_sql | Позволяет указать, требуется ли логировать все SQL-выражения на консоль. |
| mapping | Позволяет указать класс, отображающий таблицу из базы данных. |

Пример файла конфигурации Hibernate представлен на рисунке 3.5.

```
<?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:5432/auction</property>
        <property name="connection.driver_class">org.postgresql.Driver</property>
        <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
        <property name="hibernate.connection.username">postgres</property>
        <property name="hibernate.connection.password">root</property>
        <property name="hibernate.temp.use_jdbc_metadata_defaults">>false</property>
        <property name="hibernate.show_sql">>false</property>
        <!--<property name="transaction.auto_close_session">>false</property>-->
        <!--<property name="hbm2ddl.auto">update</property>-->

        <mapping class="by.iba.uzhyhala.entity.AddressEntity"/>
        <mapping class="by.iba.uzhyhala.entity.AuthInfoEntity"/>
        <mapping class="by.iba.uzhyhala.entity.BetEntity"/>
        <mapping class="by.iba.uzhyhala.entity.CategoryEntity"/>
        <mapping class="by.iba.uzhyhala.entity.LotEntity"/>
        <mapping class="by.iba.uzhyhala.entity.FeedbackEntity"/>
        <mapping class="by.iba.uzhyhala.entity.PersonalInformationEntity"/>
    </session-factory>
</hibernate-configuration>
```

Рисунок 3.5 – Скриншот исходного код файла hibernate.cfg.xml

Сгенерировать класс, отражающий таблицу в базе данных. Для хранения истории торгов был использован класс – BetEntity, приведенный на рисунке 3.6

```
@Entity
@Table(name = "bet", schema = "public", catalog = "auction")
public class BetEntity implements Serializable {
    private static final long serialVersionUID = -4773465060718156001L;
    private int id;
    private String uuid;
    private String bulk;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    @Basic
    @Column(name = "uuid", nullable = false, length = -1)
    public String getUuid() {
        return uuid;
    }
    public void setUuid(String uuid) {
        this.uuid = uuid;
    }
    @Basic
    @Column(name = "bulk", nullable = true, length = -1)
    public String getBulk() {
        return bulk;
    }
    public void setBulk(String bulk) {
        this.bulk = bulk;
    }
}
```

Рисунок 3.6 – Скриншот исходного кода класса BetEntity

Для связи полей сущности и столбцов таблицы используются аннотации. Java-аннотация — в языке Java специальная форма синтаксических метаданных, которая может быть добавлена в исходный код.

Первая аннотация @Entity указывает на то, что класс отображает таблицу из базы данных.

Аннотация @Table(name = "bet", schema = "public", catalog = "auction"), говорит о том с какой таблицей будет связан класс, если название таблицы и класса не совпадает, то надо явно указать название таблицы (name = "bet").

Аннотация @Id является обязательной, требование фреймворка Hibernate — наличие поля id для каждой таблицы.

Аннотация `@GeneratedValue(strategy = GenerationType.IDENTITY)` говорит о том, что значение поля к которому она относится, будет генерироваться автоматически. В данном случае она относится к полю `id`.

Аннотация `@Column(name = "id", nullable = false)` говорит о связи поля базы данных с полем Java класса. Аргумент `nullable = false` – означает, что в таблице этот столбец не может быть пустым.

3.4 Реализация отправки писем, подключение к почтовому сервису

Для работы с почтой был создан класс `MailUtil`. Отправка писем осуществляется по протоколу SMTP.

SMTP (Simple Mail Transfer Protocol) – это протокол предназначенный для передачи электронной почты в сетях TCP/IP. Настройка параметров для отправки электронной почты, представлена на рисунке 3.7. В классе реализовано 3 метода для отправки писем:

- `sendSimpleHtmlMail` – метод, позволяет отправлять письма HTML-формата;
- `sendSimplePlainMail` – метод, позволяет отправлять письма plain-формата (обычный текст);
- `sendErrorMail` – метод, предназначенный для отправки писем содержащих описание ошибок, или же `stack trace`;
- `sendForgetPasscodeMail` – метод, реализующий отправку на почту ссылки для восстановления пароля.

```
Properties props = new Properties();
props.put("mail.smtp.host", EMAIL_HOST);
props.put("mail.smtp.port", EMAIL_PORT);
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.starttls.enable", "true");

Authenticator auth = new Authenticator() {
    @Override
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(EMAIL_SUPPORT, EMAIL_SUPPORT_PASSCODE);
    }
};
```

Рисунок 3.7 – Скриншот фрагмента класса `MailUtil`

В `Properties` мы указываем свойства, для подключения к серверу электронной почты, а именно:

- `mail.smtp.host` – указываем хост, к которому требуется подключиться, для отправки письма;
- `mail.smtp.port` – указываем порт, к которому требуется подключиться;
- `mail.smtp.auth` – указываем, требуется ли авторизация на сервере, для отправки почты;
- `mail.smtp.starttls.enable` – указываем, использовать ли защищенное подключение по TLS.

3.5 Реализация создания PDF отчета

Для создания PDF документа, использовалась библиотечка iText. iText – это библиотека, предназначенная для создания, изменения и анализа документов в формате PDF, также ее можно использовать для создания XML, HTML, RTF. Для подключения библиотеки к проекту необходимо указать зависимость, представленную на рисунке 3.8 в файле pom.xml.

```
<dependency>
    <groupId>com.itextpdf</groupId>
    <artifactId>itextpdf</artifactId>
    <version>5.0.6</version>
</dependency>
```

Рисунок 3.8 – Скриншот зависимости, для подключения библиотеки iText

Листинг исходного кода, демонстрирующий создание PDF документа представлен в приложении А. Пример документа будет приведен в руководстве пользователя.

3.6 Реализация создания Excel отчета

Для создания Excel документа, использовалась библиотека Apache POI. Apache POI – это библиотека предназначенная для работы с документами пакета Microsoft Office. Библиотека имеет широкие возможности, а именно чтение и запись файла, также чтение файлов с паролем. Исходный код метода для создания Excel документа приведен на рисунке 3.9.

```
public static Workbook createExcelFile(List<Map<String, String>> dataList,
                                       List<String> columnList, String sheetName) {
    Workbook workbook = new XSSFWorkbook();
    Sheet sheet = workbook.createSheet(sheetName);
    Row rowHeader = sheet.createRow(0);

    for (int i = 0; i < columnList.size(); i++) {
        Cell cell = rowHeader.createCell(i);
        cell.setCellValue(valueOf(columnList.get(i)));
    }

    int rowNumber = 1;
    for (Map<String, String> stringMap : dataList) {
        Row row = sheet.createRow(rowNumber++);
        int columnNumber = 0;
        for (String column : columnList) {
            Cell cell = row.createCell(columnNumber++);
            cell.setCellValue(stringMap.get(column));
        }
    }
    return workbook;
}
```

Рисунок 3.9 – Скриншот метода создания Excel документа

Для подключения библиотеки к проекту необходимо указать зависимости, представленные на рисунке 3.10 в файле pom.xml.

```
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi</artifactId>
  <version>${poi.version}</version>
</dependency>
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi-ooxml</artifactId>
  <version>${poi.version}</version>
</dependency>
```

Рисунок 3.10 – Скриншот зависимостей, для подключения библиотеки Apache POI

Библиотека позволяет работать со следующими типами файлов: Microsoft Excel, Office Open XML, Microsoft Word, Microsoft PowerPoint, Microsoft Visio, Publisher, Publisher.

3.7 Реализация механизма проведения торгов

Для реализации механизма хранения и проведения торгов используется JSON строка представленная на рисунке 3.11. При создании нового лота, автоматически генерируется пустая JSON строка, которая хранится в поле bulk таблица bet.

После принятия ставки происходит изменения JSON строки, куда дописывается информация о размере ставки, времени, пользователе сделавшего ставку. Для работы с JSON-строкой использовалась библиотека GSON, компании Google [18].

```
{
  "uuid_lot": "1caf7993-68b4-4475-a642-206d3413d469",
  "uuid_seller": "14209d9b-dc3a-4daa-9fe9-6d056feb3af",
  "uuid_client": "",
  "status": "active",
  "blitz_cost": 10000,
  "step": 200,
  "bets": [
    {
      "uuid_user": "14209d9b-dc3a-4daa-9fe9-6d056feb3af",
      "uuid_bet": "46daf748-4736-487e-a08e-d651a9885ecd",
      "bet": 0,
      "old_cost": 100,
      "new_cost": 100,
      "date": "10-04-2018",
      "time": "09:24:46:103"
    }
  ]
}
```

Рисунок 3.11 – Скриншот структуры JSON строки, в поле bulk

Описание полей и массивов JSON строки приведено ниже:

- uuid_lot – хранит уникальный идентификатор лота;
- uuid_seller – хранит идентификатор пользователя создавшего лот;

- `uuid_client` – хранит уникальный идентификатор пользователя, победившего в торгах. В случае если торги не состоялись, поле остается пустым;
- `status` – хранит статус лота и можешь одержать один из параметров:
- `blitz_cost` – хранит блиц цену на лот, указывается пользователем при добавлении лота;

- `step` – хранит шаг цены, минимальная ставка не может быть меньше.

Для хранения истории ставок на лот используется массив, в который добавляются ставки в ходе аукциона. Описание массива `bets` приводится ниже.

Каждый элемент массива содержит следующие поля:

- `uuid_user` – хранит идентификатор пользователя сделавшего ставку;
- `uuid_bet` – уникальный идентификатор ставки;
- `bet` – размер ставки, при создании лота первая ставка ставка 0 рублей, автоматически указывается от имени пользователя, создавшего лот;
- `old_cost` – старая цена за лот, в случае если ставок еще нет, указывается начальная цена за лот;

- `new_cost` – новая цена за лот, в случае если ставок еще нет, указывается начальная цена за лот;

- `date` – дата, когда была сделана ставка;

- `time` – время, когда была сделана ставка.

Столбец `status` может содержать следующие статусы:

- `active` – аналогично «Доступен для ставок». Лот, помеченный данным статусом, участвует в торгах в настоящий момент;

- `sales` – аналогично «Продано». Лот, помеченный данным статусом, уже продан, ставки не принимаются;

- `close` – аналогично «Закрыт». Лот, помечается данным статусом если торги не состоялись;

- `wait` – аналогично «В ожидании». Лот, помеченный данным статусом, добавлен в базу и ожидает своего времени начало торгов.

Метод генерации JSON-строки представлен на рисунке 3.12.

```
private String prepareBetBulk(String uuidUser, String uuidLot, String status, String cost, String blitz, String step) {
    return "{\n" +
        "  \"uuid_lot\": \"" + uuidLot.trim() + NEW_LINE +
        "  \"uuid_seller\": \"" + uuidUser.trim() + NEW_LINE +
        "  \"uuid_client\": \"\", \n" +
        "  \"status\": \"" + status.trim() + NEW_LINE +
        "  \"blitz_cost\": \"" + Integer.parseInt(blitz.trim()) + NEW_LINE +
        "  \"step\": \"" + Integer.parseInt(step.trim()) + NEW_LINE +
        "  \"bets\": [\n" +
        "    {\n" +
        "      \"uuid_user\": \"" + uuidUser.trim() + NEW_LINE +
        "      \"uuid_bet\": \"" + UUID.randomUUID().toString() + NEW_LINE +
        "      \"bet\": 0, \n" +
        "      \"old_cost\": " + Integer.parseInt(cost.trim()) + ", \n" +
        "      \"new_cost\": " + Integer.parseInt(cost.trim()) + ", \n" +
        "      \"date\": \"" + new SimpleDateFormat(PATTERN_DATE).format(new Date().getTime()).trim() + NEW_LINE +
        "      \"time\": \"" + new SimpleDateFormat(PATTERN_TIME_WITH_MILLISECONDS).format(new Date().getTime()).trim() + "\" \n" +
        "    } \n" +
        "  ] \n" +
        "}";
}
```

Рисунок 3.12 – Скриншот метода генерации JSON-строки, при добавлении лота

Для того чтобы сделать ставку, пользователь должен иметь учетную запись в системе и быть авторизован. После этого пользователь может участвовать в торгах. Если статус лота «Доступен для ставок», то пользователь может сделать ставку не ниже минимального шага или поставить цену больше или равную блиц-цене, если блиц цена не достигнута ставками менее заявленной блиц-цены.

Если торги состоялись, пользователь продавец и пользователь, который победил получают сообщение по электронной почте об успешно завершенных торгах. Также пользователю будет отправлены контакты продавца для связи.

Если торги не состоялись, пользователь разместивший лот, получит уведомление по почте, о том, что статус лота был изменен. Пользователь будет иметь возможность повторю выставить на торги лот или же удалить его из системы. Скриншот, отражающая логику проведения торгов отображен на рисунке 3.2 [8].

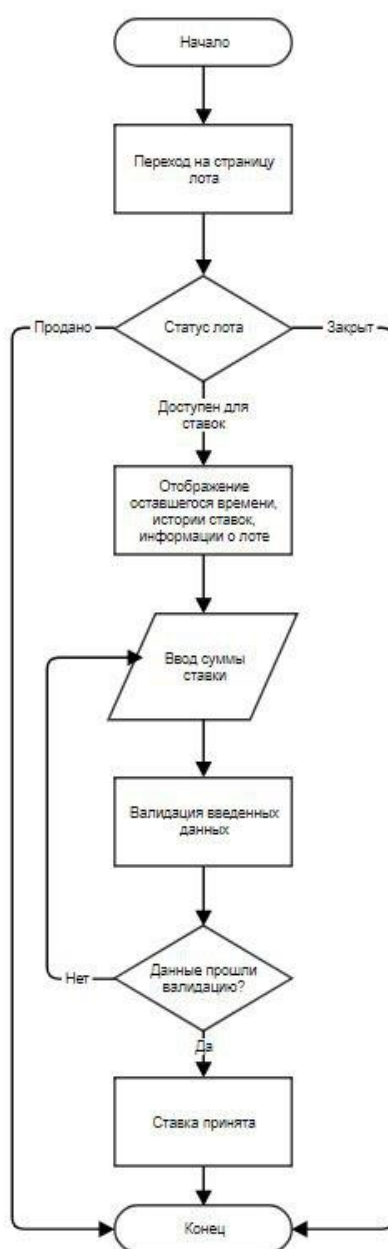


Рисунок 3.13 – Скриншот схемы алгоритма ставки

На блок-схеме представлена логика ставки на лот, статус «В ожидании» в блок-схеме не рассматривается, он аналогичен по логике «Закрит» и «Продано».

3.8 Реализация API

API (Application Programming Interface) – это интерфейс взаимодействия между сайтом и сторонними программами и серверами. API позволяет сторонним разработчикам получить доступ к функционалу сервиса предоставляющего API. Диаграмма последовательность для API запроса представлена на рисунке 3.14.

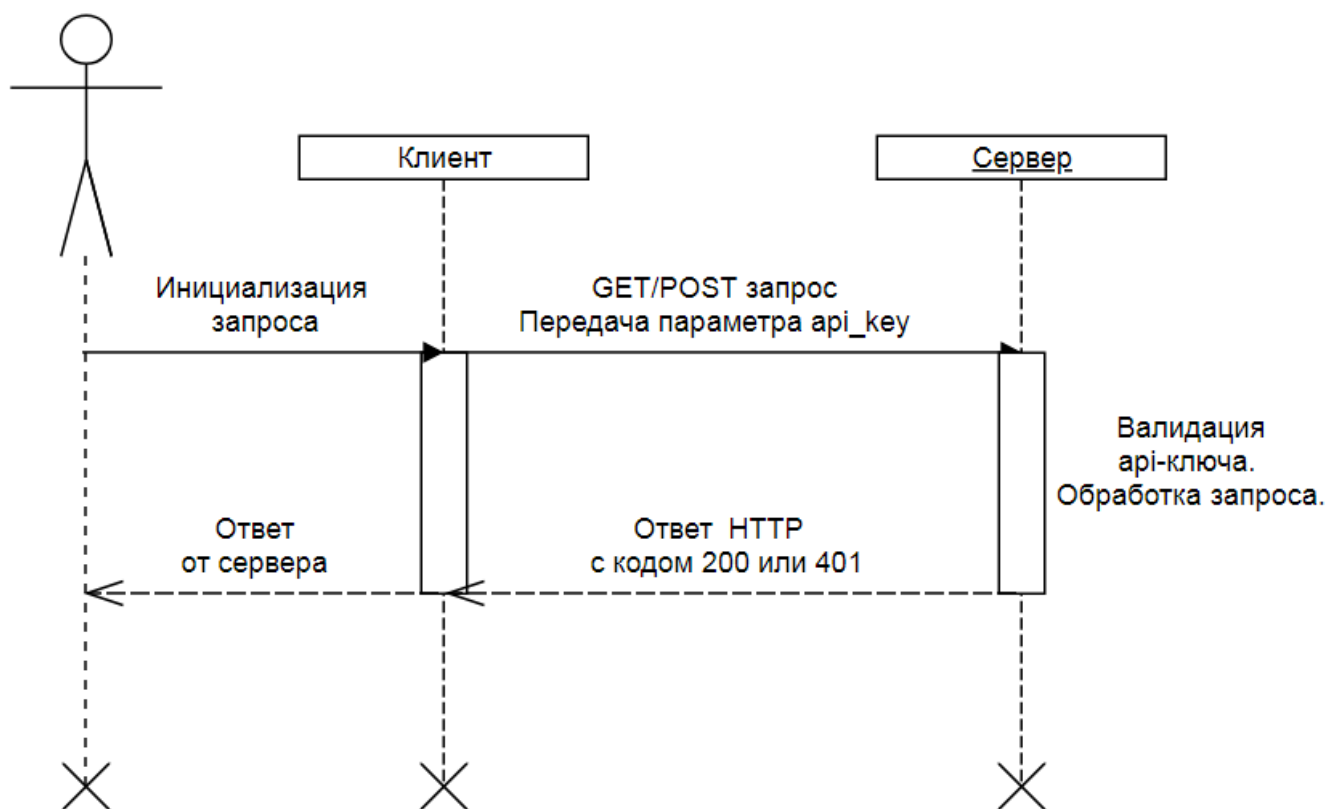


Рисунок 3.14 – Диаграмма последовательности для API запроса

Из диаграммы последовательности мы видим, что пользователь инициирует запрос по API, передает его GET или POST методом и в запрос включается `api_key`, если это требуется. Ключ проверяется на стороне сервера и есть валидация пройдена, сервер обрабатывает запрос и возвращает ответ клиенту.

Ответ от сервера – JSON строка. Краткое описание API запросов, реализованных в данном программном средстве приведено в таблице 3.12.

Таблица 3.12 – API предоставляемое программным средством

| URI | HTTP метод | Описание параметров | Параметры запроса | Назначение |
|---------------|------------|-------------------------------------|---|--------------------------|
| /api/user/reg | POST | Параметры передаются в теле запроса | <pre>{ "login": "test", "password": "test", "email": "test@test.test" }</pre> | Регистрация пользователя |

Продолжение таблицы 3.12

| | | | | |
|---------------------------|------|--|--|--|
| /api/lot/add | POST | Параметры передаются в теле запроса, api_key – уникальный ключ для работы с API | { "uuid_user_seller": "87ff415e-b8ea-481b-964d-c23815e97cb5", "name": "Lot name", "information": "info lot", "cost": "1000", "blitz_cost": "10000", "step_cost": "500", "date_start": "2018-04-22", "time_start": "00:00", "id_category": 1 } | Добавление лота |
| /status | GET | uuid – уникальный идентификатор лота | uuid-lot= 5c6b7e7c-aa1a-4e9a-94eb-31e9fe81f4c2 &api_key= D2B331E0831A4C5683E17FDA0394723C | Изменение статуса лота |
| /api/lot/infobet | GET | uuid – уникальный идентификатор лота, api_key – уникальный ключ для работы с API | uuid=fb56af7f-4af0-4d2c-9175-61d6ae7b50c7 &api_key=D2B331E0831A4C5683E17FDA0394723C | Получение информации ставок на лоте |
| /del | GET | uuid-lot – уникальный идентификатор лота, api_key – уникальный ключ для работы с API | uuid-lot=5c6b7e7c-aa1a-4e9a-94eb-31e9fe81f4c2 &api_key=D2B331E0831A4C5683E17FDA0394723C | Удаление лота |
| /api/document/bet-history | GET | uuid – уник. идентификатор лота, api_key – уник. ключ для работы с API type – тип документа | uuid=d83a7aa9-a099-46e1-94a9-af145ac54b8e &api_key=D2B331E0831A4C5683E17FDA0394723C&type=(pdf, excel) | Получение документа с историей ставок на лот |

Продолжение таблицы 3.12

| | | | | |
|------------|-----|--|---|-----------------------------------|
| /getapikey | GET | uuid – уникальный идентификатор пользователя | uuid=7418fe43-6076-418d-9daa-f3e4f9cd4049 | Получить api key для пользователя |
|------------|-----|--|---|-----------------------------------|

Рассмотрим запрос к серверу для получения истории ставок на лот. Отправим GET-запрос по адресу: http://localhost:8080/api/lot/infobet?uuid=fb56af7f-4af0-4d2c-9175-61d6ae7b50c7&api_key=D2B331E0831A4C5683E17FDA0394723C.

В строке запроса передаем два параметра:

- uuid – уникальный идентификатор лота, для которого требуется получить историю ставок.

- api_key – уникальный ключ для использования API. Ключ запрашивается пользователем в профиле и однозначно идентифицирует пользователя, использующего API.

Ответ от сервера приходит в формате JSON. Данный формат ответа был выбран по причине того, что с ним удобно работать и он не избыточен как формат XML. Пример ответа от сервера приведен на рисунке 3.15.

```
{
  "uuid_lot": "fb56af7f-4af0-4d2c-9175-61d6ae7b50c7",
  "uuid_seller": "87ff415e-b8ea-481b-964d-c23815e97cb5",
  "uuid_client": "",
  "status": "active",
  "blitz_cost": "10000",
  "step": "500",
  "bets": [
    {
      "uuid_user": "87ff415e-b8ea-481b-964d-c23815e97cb5",
      "uuid_bet": "97e77e9d-0542-4c34-9fff-17f20952bcaf",
      "bet": 0,
      "old_cost": 1000,
      "new_cost": 1000,
      "date": "01-05-2018",
      "time": "20:59:00"
    },
    {
      "uuid_user": "e3b59352-4daa-4e21-85b7-adc49d4df54c",
      "uuid_bet": "f502322d-05de-43d4-994e-0e355a5b7706",
      "bet": 1000,
      "old_cost": 1000,
      "new_cost": 2000,
      "date": "01-05-2018",
      "time": "21:05:09"
    }
  ]
}
```

Рисунок 3.15 – Скриншот ответа от сервера на API запрос /api/lot/infobet

На рисунке 3.15 приведен – ответ от сервера. На данном листинге мы видим, что на лот было произведено 2 ставки, статус лота «active» - доступен для ставок. Пример класса, обрабатывающего API запрос представлен на рисунке 3.16.

```

@WebServlet(urlPatterns = "/api/lot/infobet")
public class LotInfoBetAPI extends HttpServlet {
    private static final Logger LOGGER = Logger.getLogger(LotInfoBetAPI.class);
    private static final long serialVersionUID = -5847046564494053976L;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) {
        resp.setContentType("application/json");
        resp.setCharacterEncoding("UTF-8");
        String message;
        String uuidLot = req.getParameter("uuid");
        try {
            if (isApiKeyValid(req.getParameter(PARAMETER_API_KEY_NAME))) {
                LOGGER.info("uuid lot: " + uuidLot + ", api_key: " + req.getParameter(PARAMETER_API_KEY_NAME));
                String bulk = getJsonBetBulk(uuidLot);
                if (!StringUtils.isBlank(bulk))
                    message = bulk;
                else
                    message = "{\"uuid_lot\": \"" + uuidLot + "\", \"exception\": \"Info not found, please check uuid lot\"}";
            } else {
                message = "{\"exception\": \"Api key isnt valid\"}";
            }
            resp.getWriter().write(message);
        } catch (Exception ex) {
            try {
                resp.getWriter().write("{\"exception\": \"" + ex.getLocalizedMessage() + "\"}");
            } catch (Exception e) {
                LOGGER.error(e.getLocalizedMessage());
                new MailUtil().sendErrorMail(Arrays.toString(e.getStackTrace()));
            }
        }
    }
}

```

Рисунок 3.16 – Скриншот класса, обрабатывающего запрос /api/lot/info

В примере приведен класс LotInfoBetAPI, который принимает GET-запрос от клиента, и возвращает информацию о лоте, запрошенном пользователем.

3.9 Реализация шифрования (хэширования) данных

Для реализации шифрования данных в программном средстве, было принято решение хэшировать пароли с использованием алгоритма sha512. Для надежности хэширования, защиты от подбора паролей применяются «соль»-слово, которое дописывается к паролю на этапе хэширования [17].

Алгоритм применения хэш-функции в программном средстве, следующий:

- пользователь создает учетную запись;
- пароль, введенный пользователем, обрабатывается алгоритмом хэширования sha512 и сохраняется в базу;
- при авторизации пользователя пароль, введенный пользователем, обрабатывается тем же алгоритмом и сверяется с хэш-кодом из базы.

3.10 Реализации модуля формирования статистики в виде графиков

Статистика для администратора будет представлена на графиках. При построении графического отображения информации необходимо соблюдать ряд требований. Прежде всего, графики должны быть наглядными и понятными, легко

восприниматься. Применение графиков упрощает восприятие информации, и позволяет визуально сравнивать информацию, отраженную на нескольких графиках или на одном, с применением нескольких линий.

Для реализации графического отображения статистики применена JavaScript библиотека Google Charts Tools API. Данная библиотека позволяет строить графики в виде гистограмм, диаграмм, деревьев, линейных графиков.

Пример графика зависимости числа добавленных лотов от даты, приводится на рисунке 3.17.

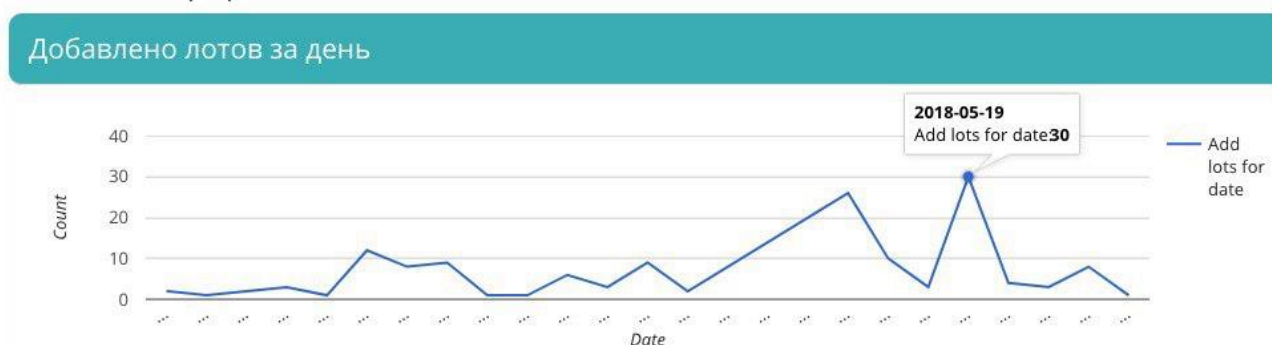


Рисунок 3.17 – Скриншот графика, построенного с применением Google Charts Tools API

Пример скрипта для построения графика представлне на рисунке 3.18.

```
<script>
google.charts.load('current', {packages: ['corechart', 'line']});
google.charts.setOnLoadCallback(lineLotAddChart);

function lineLotAddChart() {
    let data = new google.visualization.DataTable();
    data.addColumn('string', 'Date');
    data.addColumn('number', 'Add lots for date');
    data.addRows([<%=lineChartAddDateLot%>]);
    data.sort({column: 0, asc: true});
    let options = {
        hAxis: {title: 'Date'},
        vAxis: {title: 'Count'}
    };
    let chart = new google.visualization.LineChart(document.getElementById('chart_add_date_lot'));
    chart.draw(data, options);
}
</script>
```

Рисунок 3.18 – Скриншот скрипта отображающий график на странице

На рисунке 3.18 видим, что график будет отображать ломанную прямую. Зависимость переменных на графике: количество лотов от даты, сортировка отображаемых данных на графике произведена по дате и возрастанию. Формат данных для графика – строка формата [дата, количество], [дата, количество]. Скриншот метода, подготавливающий данные в нужный формат для графика представлен на рисунке 3.19.

```

public String prepareChartDataFormat(String query, String type) {    // return [date, count]
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        session.beginTransaction();

        LOGGER.debug("prepareChartDataFormat query: " + query);
        LOGGER.debug("prepareChartDataFormat type: " + type);
        List<Object[]> list = session.createQuery(query).list();
        StringBuilder dataForChart = new StringBuilder();
        for (Object[] record : list) {
            if (!valueOf(record[0]).equals("null")) {
                if (type.equals(LOT))
                    dataForChart.append("[\""].append(changeDateFormat(record[0])).append("\",").append(record[1]).append("\",");
                if (type.equals(USER))
                    dataForChart.append("[\""].append(record[0]).append("\",").append(record[1]).append("\",");
            }
        }
        String chartData = dataForChart.substring(0, dataForChart.length() - 1);
        LOGGER.debug(chartData);
        return chartData;
    } catch (Exception ex) {
        new MailUtil().sendErrorMail(Arrays.toString(ex.getStackTrace()));
        LOGGER.error(ex.getStackTrace());
        throw new IllegalArgumentException("Cannot get data");
    }
}

```

Рисунок 3.19 – Скриншот метода, подготавливающий данные в требуемом формате для графика

Метод, представленный на рисунке 3.19, является универсальным, и может быть применен для любых данных.

3.11 Реализация работы с cookies

Куки (cookies) – это фрагмент данных, который хранится на стороне клиента. Применение куки позволяет сохранить данные для идентификации пользователя, а именно роли пользователя, в программном средстве это реализовано с применением JWT-токена.

JWT-токен – это строка основанная на формате JSON, является стандартом для передачи авторизационных данных в клиент-серверных приложениях, коим является разрабатываемое программное средство.

Пример JWT-токена:

```

eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJBdXRoVG9rZW4iLCJ1dWlkIjoiZTBmMTE2MDAtZmE3NS00NGNjLTlhMDUtMDU2YmNiMTA0MmJiIiwicm9sZSI6InVzZXIifQ.tpfX_7yWIVMwA8PmH7UL2GVFoVATowk2uj6zea2iS4UCYAj-4eeKtsLQs5UrrOoY__2wIxE2GiGq6wyOSyoRbA.

```

Каждый JWT-токен состоит из трех частей: заголовок (header), полезная нагрузка или тело токена (payload), и подпись или шифрование данных. Header и payload являются JSON-объектами, а третья часть, подпись вычисляется на основании первых. В программном средстве используется шифрование токена с применением алгоритма sha512.

Вывод по разделу 3

В данном разделе рассмотрена архитектура, на которой будет построен прототип дипломного проекта, описаны пакеты и классы, которые в них содержатся.

Обоснован выбор языка программирования и СУБД, используемая в программном средстве. Изучены особенности взаимодействия с базой данных посредством ORM-подхода с использованием Hibernate. Создана база данных, которая позволит хранить данные необходимые для работы программного средства.

Хранение истории ставок, было решено хранить с применением JSON строки. Передача авторизационной информации была решена с применением JWT-токена. Для построения графиков использовалась Google Charts Tools API.

Разработано веб-приложение «Аукцион», предназначенное для проведения торгов, позволяющее разместить на торги товар, а также в удобной форме проводить торги (производить ставки), для администратора реализована возможность мониторинга веб-приложения и отображение статистики в графической форме.

Разработано API для сторонних разработчиков.

4 Тестирование веб-приложения

Пользователь ожидает, что используемое приложение простое, интуитивно понятное и работает стабильно. Исходя из ожиданий пользователя можно сделать вывод, что качество веб-приложения напрямую влияет на желание пользователя его использовать.

Качество веб-приложения – это совокупность характеристик, способных удовлетворить потребности пользователя. Дать оценку качеству веб-приложения и выявить ошибки в работе возможно с помощью этапа тестирования.

Процесс тестирования веб-приложения – это этап, на котором происходит поиск дефектов или ошибок в работе веб-приложения. В программировании есть термин – баг. Баг – используется для обозначения ошибки, и характеризуется неожиданным поведением логики веб-приложения.

Тестировщик, планируя процесс тестирования составляет тест-кейсы. Обычно тест-кейс содержит:

- описание того, что предполагается тестировать;
- последовательность действий;
- ожидаемый результат выполнения логики.

Выполняя тест-кейсы тестировщик получает результат и сравнивает его с ожидаемым результатом и, если результаты не совпали – тестировщик констатирует наличие ошибки и сообщает об этом разработчику.

Для отслеживания багов в программном средстве, часто применяются баг-трекинговые системы. После того как разработчик исправил баг, программное средство направляется на тестирование, тестировщик снова проходит шаги, требуемые в тест-кейсе.

Для ускорения процесса тестирования применяется автоматизированное тестирование. Автоматизированное тестирование – это процесс тестирования веб-приложения, при котором тестирование логики производится в автоматическом режиме с помощью инструментов для автоматизированного тестирования.

Приведем ряд моментов, которые необходимо протестировать:

- орфографические ошибки;
- контроль доступа на страницы, предназначенные только для гостя, авторизованного пользователя, администратора;
- добавления лота. Проверка добавления лота в базу, проверка вводимой информации;
- процесс торгов. Проверка логики проведения торгов (например, сумма ставки ниже заявленного шага цены);
- процессы генерации отчета. Проверка генерации отчета в PDF документ и Excel документ (например, вывод истории ставок в отчет);
- API. Проверка логики API запросов.

| | | | | | | | | | |
|-------------|------|---------------|---------|------|-----------------------------|------|--------|--|--|
| | | | | | ДП 04.00. ПЗ | | | | |
| | | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | | | | | |
| Разраб. | | Жигало В.Ю. | | | Тестирование веб-приложения | | | | |
| Провер. | | Наркевич А.С. | | | | | | | |
| Консультант | | Наркевич А.С. | | | | | | | |
| Н. Контр. | | Жиляк Н.А. | | | | | | | |
| Утверд. | | Пацей Н.В. | | | | | | | |
| | | | | | Лит. | Лист | Листов | | |
| | | | | | У | 1 | 9 | | |
| | | | | | БГТУ 74121013, 2018 | | | | |

Для отображения тестируемых элементов был составлена таблица 4.1 тест-кейсов.

Таблица 4.1 – Описание функциональных тест-кейсов

| Описание теста | Ожидаемый результат | Статус |
|--|--|---------|
| Развертывание веб-приложения | Программное средство развернуто на сервере Apache Tomcat | Успешно |
| Переход на главную страницу в роли гостя | Главная страница отображена | Успешно |
| Переход на страницу регистрации/авторизации | Страница регистрации/авторизации отображена | Успешно |
| Регистрация в программном средстве | Учетная запись пользователя создана | Успешно |
| Авторизация в программном средстве | Пользователь авторизован в системе | Успешно |
| Переход на страницу профиля | Страница профиль отображена | Успешно |
| Заполнение информации о себе на странице пользователя | Информация о пользователе обновлена | Успешно |
| Переход на страницу добавления нового лота | Страница добавления лота отображена | Успешно |
| Валидация вводимых данных о лоте | Введенные данные прошли валидацию | Успешно |
| Загрузка изображения лота | Изображение загружено | Успешно |
| Добавление лота в систему | Лот добавлен | Успешно |
| Изменение рейтинга лота (плюс) | Рейтинг изменен | Успешно |
| Изменение рейтинга лота (минус) | Рейтинг изменен | Успешно |
| Добавить ставку больше либо равную минимальному шагу цены | Ставка принята успешно | Успешно |
| Добавить ставку равную либо больше блиц-цены, если суммарная цена лота не превышает и не равна блиц-цене | Ставка принята. Статус лота изменен на продан. Пользователь получил уведомление о победе на торгах | Успешно |
| Добавить ставку меньше минимального шага цены | Ставка не принята, пользователь получил уведомление, что ставка не принята | Успешно |
| Генерация отчета истории ставок в PDF и скачать с сервера | Отчет сгенерирован, предлагается сохранить документ на локальный диск | Успешно |

Продолжение таблицы 4.1

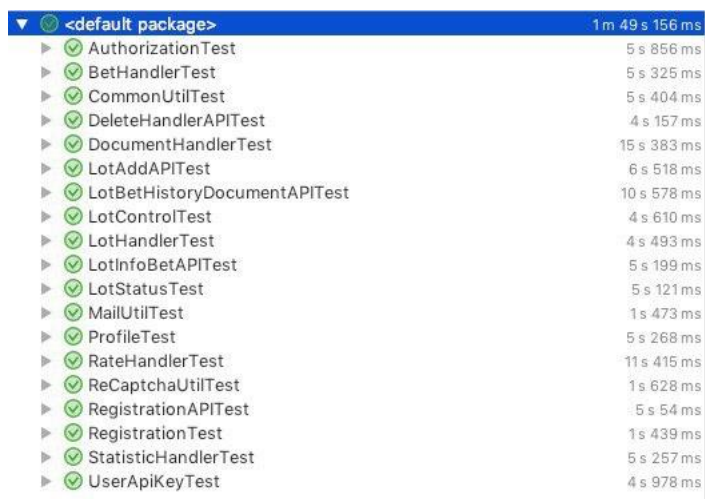
| | | |
|---|--|---------|
| Генерация отчета истории ставок в PDF и отправка на почту пользователя, указанную в профиле и/или при регистрации | Отчет сгенерирован, письмо успешно отправлено. Пользователь получит уведомление об отправке письма | Успешно |
| Генерация отчета истории ставок в Excel и отправка на почту пользователя, указанную в профиле и/или при регистрации | Отчет сгенерирован, письмо успешно отправлено. Пользователь получит уведомление об отправке письма | Успешно |
| Вход в панель администратора, с использованием логина и пароля администратора (создается при первом запуске) | Вход в панель администратора. | Успешно |
| Просмотр графиков в панели администратора | Графики отображены. Графики отображают корректную информацию | Успешно |
| Просмотр данных в таблицах (лоты, пользователи) | Таблицы отображены. Таблица отображают корректные данные | Успешно |
| Удаление пользователя или лота из панели администратора | Пользователь или лот удален | Успешно |
| Изменение информации о пользователе или лоте | Информация о пользователе или лоте изменена | Успешно |
| Генерация отчета со страницы администратора | Отчет сгенерирован. | Успешно |
| Ввод заведомо неверных данных, для тестирования отправки письма администратору с ошибкой | Исключительная ситуация создана, письмо отправлено администратору | Успешно |

В таблице, приведенной выше, дано описание функциональных тестов, проведенных на этапе тестирования.

4.1 Модульное тестирование

Модульное тестирование (unit тестирование) – это процесс тестирования веб-приложения, позволяющий проверить отдельные части исходного кода. Цель преследуемая при проведении модельного тестирования это изолировать отдельные части исходного кода и показать, что тестируемые часть кода работают логически верно.

Результат успешной работы unit-тестов показан на рисунке 4.1. Мы видим, что прохождение всех тестов занято 1 минуту 49 секунд.



| <default package> | 1 m 49 s 156 ms |
|------------------------------|-----------------|
| AuthorizationTest | 5 s 856 ms |
| BetHandlerTest | 5 s 325 ms |
| CommonUtilTest | 5 s 404 ms |
| DeleteHandlerAPITest | 4 s 157 ms |
| DocumentHandlerTest | 15 s 383 ms |
| LotAddAPITest | 6 s 518 ms |
| LotBetHistoryDocumentAPITest | 10 s 578 ms |
| LotControlTest | 4 s 610 ms |
| LotHandlerTest | 4 s 493 ms |
| LotInfoBetAPITest | 5 s 199 ms |
| LotStatusTest | 5 s 121 ms |
| MailUtilTest | 1 s 473 ms |
| ProfileTest | 5 s 268 ms |
| RateHandlerTest | 11 s 415 ms |
| ReCaptchaUtilTest | 1 s 628 ms |
| RegistrationAPITest | 5 s 54 ms |
| RegistrationTest | 1 s 439 ms |
| StatisticHandlerTest | 5 s 257 ms |
| UserApiKeyTest | 4 s 978 ms |

Рисунок 4.1 – Скриншот результата работы unit-тестов

Для тестирования разрабатываемого веб-приложения были написаны unit-тесты. Использовались библиотеки JUnit и PowerMock.

JUnit – это библиотека предназначенная для модульного тестирования исходного кода на языке Java.

PowerMock – это библиотека, позволяющая разрабатывать модульные тесты, она же работает в связке с библиотекой JUnit. PowerMock позволяет создать заглушку вызова метода и вернуть значение, как если бы метод действительно отработал. Для того чтобы использовать возможности библиотеки, необходимо аннотировать тест класс анотацией: `@RunWith(PowerMockRunner.class)`. Скриншот исходного кода класса `LotStatusTest` представлен на рисунке 4.2.

```

@RunWith(PowerMockRunner.class)
@PowerMockIgnore({"javax.xml.*", "org.xml.*", "org.w3c.*", "javax", "com.sun.org.apache.xerces.*"})
public class LotStatusTest {

    @Mock
    private SessionFactory sessionFactory;
    @Mock
    private Session session;
    @Mock
    private Transaction transaction;
    @Mock
    private Configuration configuration;
    @Mock
    private MockHttpServletRequest mockHttpServletRequest;
    @Mock
    private MockHttpServletResponse mockHttpServletResponse;

    @Before
    public void init() throws Exception {
        initMocks(this);
        when(sessionFactory.openSession()).thenReturn(session);
        when(session.beginTransaction()).thenReturn(transaction);
        sessionFactory = configuration.buildSessionFactory();
        when(mockHttpServletRequest.getParameter("uuid")).thenReturn("5a132e1d-10c5-486c-886b-756fb4b3a1f8");
        when(mockHttpServletResponse.getWriter()).thenReturn(new PrintWriter(new StringWriter()));
        whenNew(URL.class).withArguments(TEST_URL).thenReturn(mock(URL.class));
    }

    @Test
    public void test() {
        new LotStatus().doGet(mockHttpServletRequest, mockHttpServletResponse);
    }
}

```

Рисунок 4.2 – Скриншот исходного кода unit-тест класса `LotStatusTest`

Описание аннотаций приведено ниже:

- @RunWith (PowerMockRunner.class) – позволяет использовать библиотеку PowerMock;
- @PowerMockIgnore – аннотация позволяет игнорировать пакеты во время выполнения теста;
- @Mock – аннотация, непосредственно создающая Mock-объект, аналогична вызову кода mock(Class.class);
- @Before – аннотация позволяет определить возвращаемые значения методов, которые вызовутся в процессе unit-теста;
- @Test – «отправная точка» выполнения теста.

Для отображения покрытия кода тестами использовался плагин JaCoCo. JaCoCo – это плагин, отображающий результат работы unit-тестов, покрытие кода, а также строки кода, которые были обработаны в процессе unit-теста. Для подключения плагина к проекту, необходимо добавить код представленный на рисунке 4.3 в pom.xml.

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.1</version>
  <executions>
    <execution>
      <id>prepare-agent</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>report</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>post-unit-test</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
      <configuration>
        <dataFile>target/jacoco.exec</dataFile>
        <outputDirectory>target/jacoco-ut</outputDirectory>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Рисунок 4.3 – Скриншот исходного кода для подключения JaCoCo плагина

Плагин отображает процент покрытия кода тестами, просмотр возможен как по отдельным классам, так и по всему проекту [19] [20].

Покрытие кода – это величина, демонстрирующая процент исходного кода, выполненного тестами во время сборки проекта. Результат покрытия кода Unit-тестами приведен на рисунке 4.4. Мы видим, что покрытие кода составило 74%.

Diploma project Auction

Diploma project Auction

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|--------------------------|---------------------|------|-----------------|------|--------|------|--------|-------|--------|---------|--------|---------|
| by.iba.uzhyhala.util | | 76% | | 72% | 18 | 80 | 82 | 373 | 6 | 46 | 0 | 7 |
| by.iba.uzhyhala.lot | | 82% | | 68% | 15 | 55 | 74 | 405 | 0 | 29 | 0 | 5 |
| by.iba.uzhyhala.user | | 61% | | 41% | 10 | 25 | 49 | 139 | 4 | 19 | 1 | 4 |
| by.iba.uzhyhala.api.user | | 58% | | 50% | 8 | 15 | 25 | 66 | 0 | 7 | 0 | 2 |
| by.iba.uzhyhala.entity | | 71% | | n/a | 31 | 113 | 46 | 166 | 31 | 113 | 0 | 7 |
| by.iba.uzhyhala.api.lot | | 81% | | 62% | 15 | 35 | 21 | 122 | 0 | 14 | 0 | 3 |
| by.iba.uzhyhala.admin | | 63% | | 58% | 6 | 15 | 19 | 52 | 1 | 9 | 1 | 3 |
| by.iba.uzhyhala.api | | 44% | | 50% | 4 | 7 | 25 | 37 | 2 | 5 | 0 | 1 |
| by.iba.uzhyhala.api.to | | 44% | | n/a | 13 | 27 | 22 | 37 | 13 | 27 | 2 | 4 |
| by.iba.uzhyhala.lot.to | | 88% | | n/a | 4 | 32 | 5 | 49 | 4 | 32 | 0 | 3 |
| Total | 1 618 of 6 467 | 74% | 71 of 199 | 64% | 124 | 404 | 368 | 1 446 | 61 | 301 | 4 | 39 |

Рисунок 4.4 – Скриншот результата работы JaCoCo плагина

Также плагин позволяет просмотреть строки исходного кода, которые были затействованы в тесте (покрыты тестом) и те, которые не были задействованы в тесте. Пример метода приведен на рисунке 4.5.

```

public static boolean isApiKeyValid(String key) {
    LOGGER.info("isApiKeyValid method");
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        session.beginTransaction();
        return session
            .createQuery("SELECT a.uuid FROM " + ENTITY_AUTH_INFO + " a WHERE api_key = :key")
            .setParameter("key", key)
            .list()
            .size() > 0;
    } catch (Exception ex) {
        new MailUtil().sendErrorMail("Method: isApiKeyValid\n" + Arrays.toString(ex.getStackTrace()));
        LOGGER.error(ex.getLocalizedMessage());
        return false;
    }
}

```

Рисунок 4.5 – Скриншот покрытия строке тестом в методе isApiKeyValid

Расположение тестовых классов аналогично расположению класса, для которого разрабатывается unit-тест. Описание unit-тестов по пакетам, реализованных в веб-приложении приведено в таблице 4.2.

Таблица 4.2 – Описание unit-тестов

| Название теста | Описание |
|----------------------|---|
| RateHandlerTest | Тестирование класса RateHandler. Тестирование бизнес-логики изменения рейтинга для лота и пользователя. |
| StatisticHandlerTest | Тестирование класса StatisticHandler. Тестирование бизнес-логики формирования статистики. |
| LotAddAPITest | Тестирование класса LotAddAPI. Тестирование бизнес-логики добавления лота используя API запрос. |

Продолжение таблицы 4.2

| | |
|------------------------------|---|
| LotBetHistoryDocumentAPITest | Тестирование класса LotBetHistoryDocumentAPI. Тестирование бизнес-логики создания PDF или Excel документа используя API запрос. |
| LotInfoAPITest | Тестирование класса LotInfoAPITest. Тестирование бизнес-логики получения информации о лоте используя API запрос. |
| RegistrationAPITest | Тестирование класса RegistrationAPI. Тестирование бизнес-логики регистрации пользователя используя API запрос. |
| UserApiKeyTest | Тестирование класса UserApiKey. Тестирование бизнес-логики получения пользователем API ключа. |
| DeleteHandlerAPITest | Тестирование класса DeleteHandlerAPI. Тестирование бизнес-логики удаления лота или пользователя. |
| BetHandlerTest | Тестирование класса BetHandler. Тестирование бизнес-логики ставки на лот. |
| DocumentHandlerTest | Тестирование класса DocumentHandler. Тестирование бизнес-логики генерации документов PDF и Excel. |
| LotControlTest | Тестирование класса LotControl. |
| LotHandlerTest | Тестирование класса LotHandler. |
| LotStatusTest | Тестирование класса LotStatus. Тестирование бизнес-логики изменения статуса лота. |
| AuthorizationTest | Тестирование класса Authorization. Тестирование бизнес-логики авторизации пользователя в системе. |
| ProfileTest | Тестирование класса Profile. |
| RegistrationTest | Тестирование класса Registration. Тестирование бизнес-логики регистрации пользователя в системе. |
| MailUtilTest | Тестирование класса MailUtil. Тестирование бизнес-логики отправки электронного письма. |
| ReCaptchaUtilTest | Тестирование класса ReCaptchaUtil. Тестирование бизнес-логики работы ReCaptcha. |

В таблице выше, приведены все тест-классы, реализованные в дипломном проекте, дано краткое описание классов.

4.2 Тестирование кода при сборке артефакта

На этапе сборки проекта применено автоматическое тестирование исходного кода на наличие «нехорошего кода». «Нехорошим кодом» мы можем назвать:

- неиспользуемые импорты;
- неиспользуемые локальные переменные;
- пустые блоки try/catch/finally/switch;
- сложные if, for выражения;
- дубликаты кода.

Для проведения данного этапа тестирования применен PMD плагин. В случае если плагин находит ошибку, сборка проекта останавливается, в консоли отобразится результат сборки, и указания строк исходного кода с ошибками. Скриншот листинга кода для подключения PMD плагина к разрабатываемому проекту представлен на рисунке 4.6.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-pmd-plugin</artifactId>
  <version>3.9.0</version>
  <configuration>
    <targetJdk>1.8</targetJdk>
    <verbose>true</verbose>
  </configuration>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>check</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Рисунок 4.6 – Скриншот исходного кода для подключения PMD плагина

Результат тестирования кода при помощи PMD плагина представлен на рисунке 4.7.

```
[INFO]
[INFO] <<< maven-pmd-plugin:3.9.0:cpd-check (default-cli) < :cpd @ auction <<<
[INFO]
[INFO] --- maven-pmd-plugin:3.9.0:cpd-check (default-cli) @ auction ---
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.863 s
[INFO] Finished at: 2018-06-02T06:17:21+03:00
[INFO] Final Memory: 21M/394M
[INFO] -----
MacBook-Air-Vladimir:Diploma vladimirzhigalo$
```

Рисунок 4.7 – Скриншот терминала с результатом работы PMD плагина

Для поиска багов в исходном коде во время разработки применена утилита – FindBugs [20]. FindBugs – это статический анализатор кода, в связке с PMD плагином позволяет эффективнее определить проблемные места в коде. Утилита распространяется по лицензии GNU, что позволяет ее свободно использовать в проекте. Результат работы утилиты FindBugs приведе на рисунке 4.8.



Рисунок 4.8 – Скриншот результата работы утилиты FindBugs

Отличительной особенностью FindBugs является то, что утилита анализирует байт-код. На сегодняшний день, доступна версия 3.0.1 [20].

Выводы по разделу 4

В данном разделе приведены результаты тестирования исходного кода разрабатываемого веб-приложения. Обоснован выбор инструментов для тестирования. Описан подход к контролю качества кода. Реализовано покрытие unit-тестами бизнес-логики на 74%.

Дано описание применных плагинов и утилит для контроля за качеством кода во време разработки и контроля во время сборки и развертывания веб-приложения.

5 Руководство пользователя системы

Веб-приложение поддерживает работу трех категорий пользователей: гость, пользователь, администратор. Шапка веб-приложения страницы представлена на рисунке 5.1.



[Профиль\(qwe\)](#)

[Добавить лот](#)

[Авторизация/Регистрация](#)

[Админ](#)

[Обратная связь](#)

Рисунок 5.1 – Скриншот шапки веб-приложения

Фрагмент главной страницы, отображающий недавно добавленные лоты представлен на рисунке 5.2.

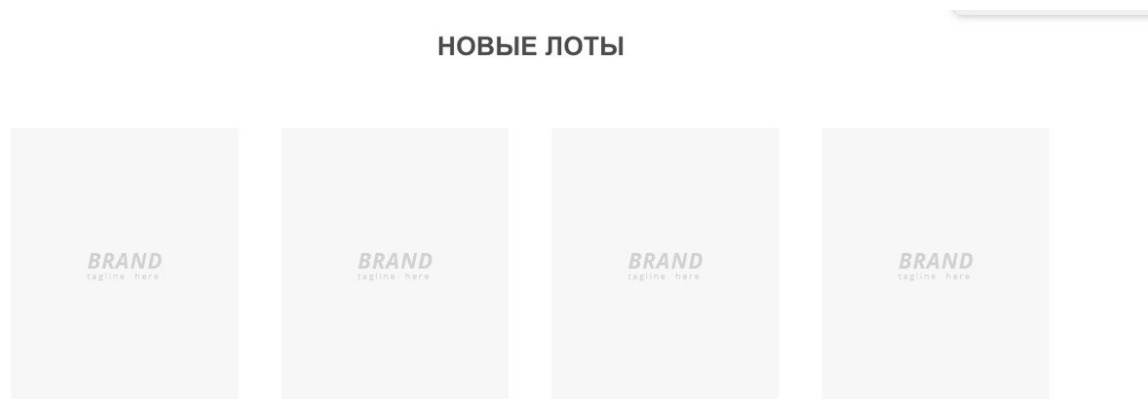


Рисунок 5.2 – Скриншот фрагмента главной страницы, отображающая недавно добавленные лоты

Краткое описание элементов приведено ниже:

- профиль – позволяет перейти к профилю пользователя, доступно только для авторизованных в системе пользователей;
- добавить лот – позволяет перейти на страницу добавления лота, доступно только для авторизованных в системе пользователей;
- авторизация/регистрация – позволяет перейти на страницу авторизации/регистрации, доступна для всех пользователей;
- админ – позволяет перейти в панель администратора. Доступна только для пользователей с ролью «Администратор». Для всех остальных пользователей переход на страницу невозможен;
- обратная связь – позволяет перейти на страницу обратной связи. Обратная связь позволяет отправить письмо администрации веб-приложения.

| | | | | | | | | | |
|-------------|------|---------------|---------|------|--|--|---------------------|------|--------|
| | | | | | ДП 05.00. ПЗ | | | | |
| | | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | Руководство пользователя веб-приложения | | Лит. | Лист | Листов |
| Разраб. | | Жигало В.Ю. | | | | | У | 1 | 8 |
| Провер. | | Наркевич А.С. | | | | | БГТУ 74121013, 2018 | | |
| Консультант | | Наркевич А.С. | | | | | | | |
| Н. Контр. | | Жуляк Н.А. | | | | | | | |
| Утверд. | | Пацей Н.В. | | | | | | | |

5.1 Гость

Гость – это пользователь, не зарегистрированный в веб-приложении, либо не прошедший авторизацию.

Для гостя доступна главная страница, страница с описанием лота, страница регистрации и авторизации рисунок, а также обратная связь рисунок. Фрагмент страницы авторизации/регистрации, предоставляющий возможность зарегистрироваться в системе, представлен на рисунке 5.3.

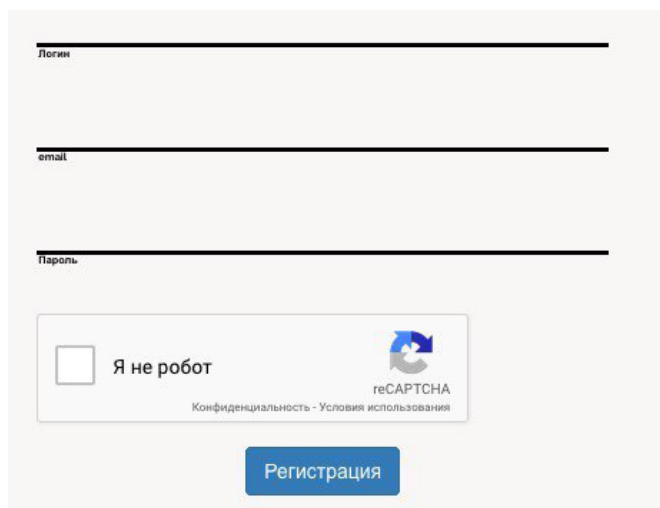
Скриншот фрагмента веб-формы. Вверху три горизонтальных поля ввода с метками 'Логин', 'email' и 'Пароль'. Ниже находится блок с капчей: слева поле с чекбоксом и текстом 'Я не робот', справа логотип reCAPTCHA и текст 'reCAPTCHA'. Под капчей мелким шрифтом написано 'Конфиденциальность - Условия использования'. В самом низу находится синяя кнопка с белым текстом 'Регистрация'.

Рисунок 5.3 – Скриншот фрагмента предоставляющий возможность зарегистрироваться в веб-приложении

Все лоты, имеющие статус активных (торги ведутся в настоящее время) доступны для просмотра незарегистрированным пользователям.

5.2 Пользователь

Пользователь – это, лицо выразившее желание взаимодействовать с веб-приложением. Фрагмент страницы авторизации/регистрации, предоставляющий возможность зарегистрироваться в системе, представлен на рисунке 5.4.

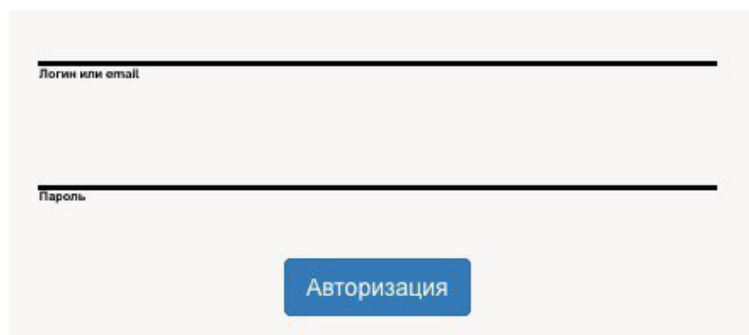
Скриншот фрагмента веб-формы. Вверху два горизонтальных поля ввода. Первое поле имеет метку 'Логин или email', второе – 'Пароль'. В самом низу находится синяя кнопка с белым текстом 'Авторизация'.

Рисунок 5.4 – Скриншот фрагмента предоставляющий возможность авторизоваться в веб-приложении

Если пользователь забыл пароль, он может запросить восстановление пароля, для этого ему надо заполнить форму, представленную на рисунке 5.5.

Рисунок 5.5 – Фрагмент модального окна для восстановления пароля

Для пользователя доступна главная страница с лотами, добавление лота, изменение и удаление своего лота, просмотр истории ставок, создание документа с историей ставок в форматах PDF или Excel, с возможностью отправки на почту или скачивания непосредственно из веб-приложения. Авторизованные пользователи имеют доступ к профилю, пример приведен на рисунке 5.6.

| Profile info | Address |
|--|----------------|
| Uuid: 87ff415e-b8ea-481b-964d-c23815e97cb5 | Страна: BY |
| Имя: Uladzimir Zhyhala | Город: Minsk |
| Дата рождения: 1996-09-29 | Улица: Kylvan |
| Рейтинг: 4 | Дом: 16 |
| | Индекс: 220100 |

Рисунок 5.6 – Скриншот информации о пользователе на странице «Профиль»

Авторизованные пользователи могут принимать участие в торгах. Пример фрагмента страницы с описанием лота представлен на рисунке 5.7.

| Информация о лоте | |
|---------------------|--------|
| Стартовая цена: | 1000 |
| Блиц цена: | 100000 |
| Минимальная ставка: | 1000 |
| Текущая цена: | 1000 |

| | |
|-----------|--------------------------------------|
| UUID: | 1a257cf7-ce7c-42cb-b6b8-8262ac1826cc |
| Название: | LOT NAME |
| Описание: | LOT INFO |
| Статус: | Доступен для ставок |
| Рейтинг: | 0 |

Рисунок 5.7 – Скриншот страницы с описание лота

Пример поля для ввода ставки приведен на рисунке 5.8. На странице «Информация о лоте» отображается оставшееся время торгов, и подсказка о минимальной цене.

Осталось времени: 09:50

Сделать ставку:

Не менее 1233

Сделать ставку

Рисунок 5.8 – Скриншот фрагмента, отображающего оставшееся время для торгов

История ставок отображается в виде таблицы, представлена на рисунке 5.9. Пользователь, победивший в торгах, выделяется цветом, если торги не состоялись – таблица ставок будет пустой.

| История ставок | | | |
|-------------------|--------|------------|----------|
| Пользователь | Ставка | Дата | Время |
| Uladzimir Zhyhala | 1500 | 29-05-2018 | 03:07:53 |
| Uladzimir Zhyhala | 1233 | 29-05-2018 | 03:07:59 |
| Uladzimir Zhyhala | 10000 | 29-05-2018 | 03:08:09 |

Рисунок 5.9 – Скриншот истории ставок на странице лота

Если торги состоялись, победитель будет объявлен на странице лота. Пример отображения победителя торгов приведен на рисунке 5.10.

Победитель: Uladzimir Zhyhala, цена: 12856

Рисунок 5.10 – Скриншот, отображение победителя и цена, за которую был приобретен товар на странице «Информация о лоте»

После успешно завершённых торгов, продавцу и покупателю будет отправлено письмо с документом подтверждающим сделку. Пример документа, подтверждающего сделку представлен на рисунке 5.11.

Auction Diploma
Sales confirmation
04/06/2018, 02:03:10



This document confirms the sale of the lot LOT NAME

----- LOT -----

Name: LOT NAME
Info: LOT INFO

----- SELLER -----

Name: Uladzimir Zhyhala
Email: qwe@qwe.qwe

----- CLIENT -----

Name: Uladzimir Zhyhala
Email: qwe@qwe.qwe

Client address
Country: BY
City: Minsk
Street: Kylman
House: 16
Zip: 220100

UUID lot: 59e3cef8-2c6a-4867-9752-47b78191b1bd
URL lot: http://localhost:8080/pages/lot.jsp?uuid=59e3cef8-2c6a-4867-9752-47b78191b1bd

Рисунок 5.11 – Скриншот документа, подтверждающего сделку

Пользователи имеют возможность скачать историю ставок в документах PDF и Excel. Пример PDF документ с историй ставок представлен на рисунке 5.12. По запросу пользователя документ может быть скачен или же отправлен на почту.

Auction Diploma
Bet history
16/05/2018, 12:32:50

Rate: 0

| First/Last name | Bet | Date | Time |
|------------------|------|------------|----------|
| Uladimir Zhyhala | 1000 | 15-05-2018 | 18:12:50 |
| Uladimir Zhyhala | 80 | 15-05-2018 | 18:12:55 |
| Uladimir Zhyhala | 123 | 15-05-2018 | 18:13:01 |
| Uladimir Zhyhala | 400 | 15-05-2018 | 18:13:11 |

UUID lot: 58b0cd13-5b4f-4983-858b-8cd11b01ecd9
URL lot: http://localhost:8080/pages/lot.jsp?uuid=58b0cd13-5b4f-4983-858b-8cd11b01ecd9




Рисунок 5.12 – Скриншот PDF документа с историей ставок

5.3 Администратор

Администратор – это пользователь, имеющий доступ к странице администратора, которая представлена на рисунке 5.13. Для администратора доступны все страницы веб-приложения, а также страница администратора. Страница администратора позволяет получить статистику об использовании веб-приложения пользователями, создать отчеты и просмотреть графики.



Рисунок 5.13 – Скриншот главной страницы панели администратора

Со стартовой страницы администратора доступен переход на страницы:

- графики;
- пользователи;
- лоты;
- Выход на основной сайт.

Также на стартовой странице отображается числовая статистика за текущий день и за все время использования веб-приложения. Статистика делится на статистику по лотам и пользователям.

На странице графики отображены графики с использованием Google Charts Tools API. Все графики представляют собой ломанную прямую, выше оси X. Графики отображают статистику по лотам и пользователям, пример графика, отображающего статистику добавленных лотов за определенный день представлен на рисунке 5.14.

Статистика по лотам отображает:

- количество добавлены за день;
- количество начатых торгов за день;
- количество состоявшихся торгов за день.

Статистика по пользователям отображает:

- количество зарегистрированных пользователей в системе по дням.

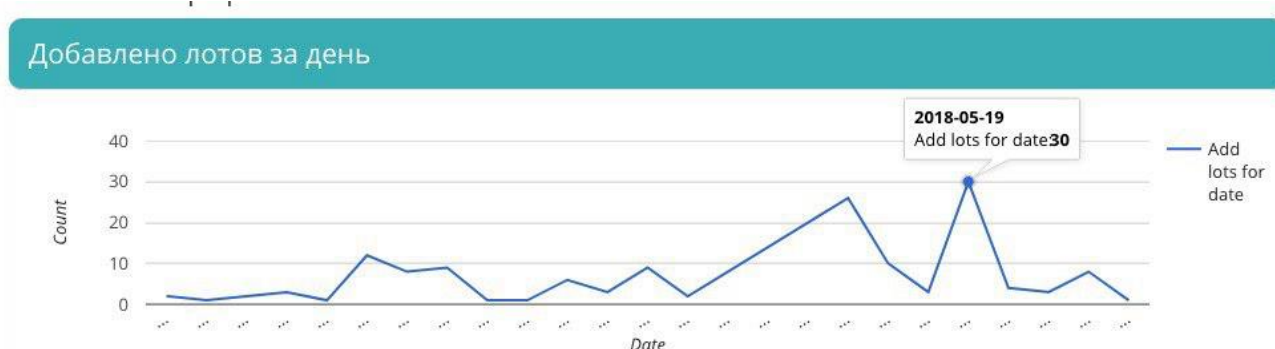


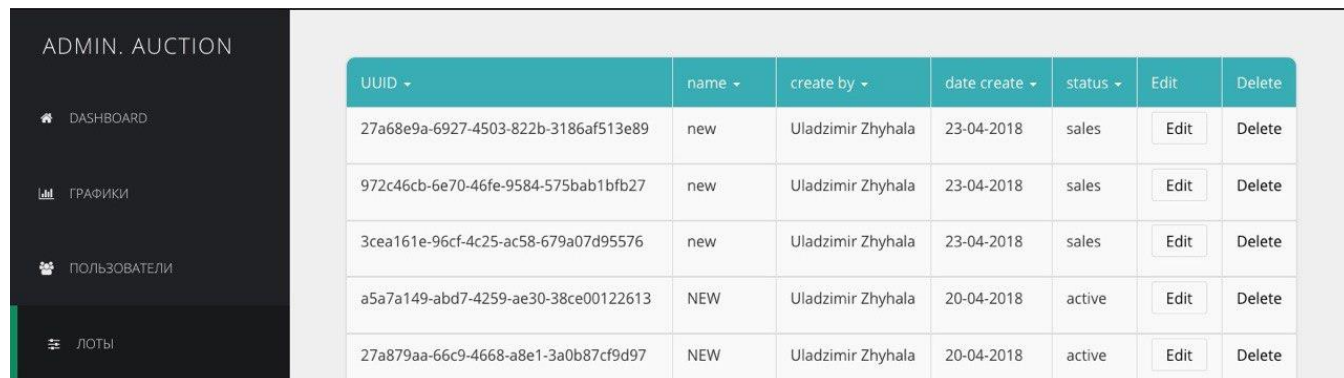
Рисунок 5.14 – Скриншот графика, отображающего количество добавленных лотов за день

На странице пользователей отображена таблица всех зарегистрированных пользователей веб-приложения. Пример страницы с таблицей пользователей представлен на рисунке 5.15. Администратор имеет возможность изменения и удаления пользователей.

| ADMIN. AUCTION DASHBOARD ГРАФИКИ ПОЛЬЗОВАТЕЛИ | | | | | | | |
|--|--------------------------------------|-------|-------------------|-------|-------------------|------|--------|
| | UUID | Login | Email | Role | Date registration | Edit | Delete |
| | 293f3466-ae17-4b47-ab64-77bf85006c1d | aaa | aaa@aaa.aaa | user | 2018-04-19 | Edit | Delete |
| | 9d2fe184-a053-4e23-b471-d9910961c6eb | ww1w | apiw11i@test.test | user | 2018-05-28 | Edit | Delete |
| | 87ff415e-b8ea-481b-964d-c23815e97cb5 | qwe | qwe@qwe.qwe | admin | 2018-04-18 | Edit | Delete |

Рисунок 5.15 – Скриншот страницы «Пользователи» в панели администратора

На странице лоты отображена таблица, содержащая все лоты, добавленные пользователями. Также с указанной страницы доступно редактирование и удаление пользователей. Пример страницы «Лоты» приведена на рисунке 5.16.



| UUID | name | create by | date create | status | Edit | Delete |
|--------------------------------------|------|-------------------|-------------|--------|------|--------|
| 27a68e9a-6927-4503-822b-3186af513e89 | new | Uladzimir Zhyhala | 23-04-2018 | sales | Edit | Delete |
| 972c46cb-6e70-46fe-9584-575bab1bfb27 | new | Uladzimir Zhyhala | 23-04-2018 | sales | Edit | Delete |
| 3cea161e-96cf-4c25-ac58-679a07d95576 | new | Uladzimir Zhyhala | 23-04-2018 | sales | Edit | Delete |
| a5a7a149-abd7-4259-ae30-38ce00122613 | NEW | Uladzimir Zhyhala | 20-04-2018 | active | Edit | Delete |
| 27a879aa-66c9-4668-a8e1-3a0b87cf9d97 | NEW | Uladzimir Zhyhala | 20-04-2018 | active | Edit | Delete |

Рисунок 5.16 – Скриншот страницы «Лоты» в панели администратора

На главной странице панели администратора, приведена числовая статистика за текущий день рисунок 5.17 и за все время рисунок 5.18.



| СТАТИСТИКА ЗА СЕГОДНЯ | |
|---|---|
| Лоты | |
| Добавлено лотов за день | 1 |
| Завершено лотов за день | 1 |
| Начато торгов за день | 1 |
| Пользователи | |
| Зарегистрировано пользователей за сегодня | 2 |

Рисунок 5.17 – Скриншот статистики за текущий день, на главной странице панели администратора

Статистика по лотам за сегодня:

- количество добавленных лотов;
- количество завершенных лотов;
- количество начатых торгов за день.

Статистика по пользователям за сегодня:

- количество зарегистрированных пользователей за текущий день.
- число лотов добавленных за текущий








| СТАТИСТИКА ЗА ВСЁ ВРЕМЯ | | |
|---|----------------------------------|-----|
| Лоты | | |
|  | Всего лотов на площадке | 187 |
|  | Всего лотов доступных для ставок | 18 |
|  | Всего лотов в ожидании торгов | 139 |
|  | Всего продано лотов | 23 |
|  | Всего отмененных лотов | 7 |
| Пользователи | | |
|  | Всего пользователей | 9 |
|  | Пользователи использующие API | 4 |

Рисунок 5.18 – Скриншот статистики за всё время, на главной странице панели администратора

Статистика по лотам за все время отображает:

- общее количество лотов на площадке;
- количество лотов со статусом «Доступен для ставок»;
- количество лотов со статусом «В ожидании»;
- количество лотов со статусом «Продано»;
- количество лотов со статусом «Закрыт».

Статистика по пользователям за всё время отображает следующее:

- общее количество пользователей, зарегистрированных в системе;
- количество пользователей с ролью «администратор»;
- количество пользователей с ролью «пользователь»;
- количество пользователей, использующих API.

Вывод по разделу 5

В данном разделе представлено руководство для пользователя по работе с веб-приложением. Она описывает основных возможности, которые, которые реализованы в веб-приложении.

Пользователю доступны регистрация и авторизация по логину или электронной почте и паролю. Авторизованный пользователь может выставить лот на торги, участвовать в торгах, получить отчет с историей ставок.

В разделе дано описание возможности взаимодействия в панели администратора. Описанные варианты отображения статистики, а также основное назначение страниц, в панели адмнистратора.

6 Экономический раздел

6.1 Общая характеристика разрабатываемого веб-приложения

Основной целью экономического раздела является экономическое обоснование целесообразности разработки веб-приложения, представленного в дипломном проекте.

В этом разделе пояснительной записки проводится расчет затрат на всех стадиях разработки, а также анализ экономического эффекта в связи с использованием данного веб-приложения.

Разработка проектов программных средств требует разнообразных затрат и не редко значительных объемов ресурсов (трудовых, материальных, финансовых). В связи с этим, разработка и реализация каждого проекта должна быть обоснована, как технически, так и экономически.

Программное средство разработано с использованием объектно-ориентированного языка Java, среды разработки IntelliJ IDEA.

6.2 Исходные данные

Исходные данные для расчета стоимости программного продукта представлены в таблице 6.1.

Таблица 6.1 – Исходные данные расчета

| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
|--|-------------------|-----------------------|----------|
| Численность разработчиков | чел. | Ч _р | 1 |
| Норматив дополнительной заработной платы | % | Н _{дз} | 10 |
| Ставка отчислений в Фонд социальной защиты населения и Белгосстрах | % | Н _{фсзн} | 34,4 |
| Цена одного машино-часа | руб. | С _{мч} | 0,05 |
| Норматив прочих затрат | % | Н _{пз} | 10 |
| Норматив накладных расходов | % | Н _{обп, обх} | 100 |
| Норматив расходов на сопровождение и адаптацию | % | Н _{рса} | 10 |
| Ставка НДС | % | Н _{ндс} | 20 |

Для проведения маркетингового анализа исследовался рынок подобных продуктов. Разработка любого веб-приложения состоит из трех этапов: дизайн, верстка, программирование. Стоимость разработки дизайна веб-приложения составляет 140 рублей за один макет.

| | | | | | | | | |
|-------------|------|---------------|---------|------|----------------------|------|--------|--|
| | | | | | ДП 06.00. ПЗ | | | |
| | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | Экономический раздел | | | |
| Разраб. | | Жигало В.Ю. | | | | | | |
| Провер. | | Наркевич А.С. | | | | | | |
| Консультант | | Евлаш А.И. | | | | | | |
| Н. Контр. | | Жиляк Н.А. | | | | | | |
| Утверд. | | Пацей Н.В. | | | БГТУ 74121013, 2018 | | | |
| | | | | | | | | |
| | | | | | Лит. | Лист | Листов | |
| | | | | | У | 1 | 9 | |

В разработанном веб-приложении отрисовано десять макетов, соответственно, общая стоимость за дизайн проекта составляет 1400 рублей. Средняя стоимость верстки 110 рублей за страницу. Все сверстано десять типовых страниц стоимостью в 1100 рублей. Стоимость программирования составляет 36 рублей за один час.

С учетом того, что проект разрабатывался три месяца (63 рабочих дня), где один день составляет 8 часов, общая стоимость программирования составляет 18144 рублей. Внутренняя SEO-оптимизация составляет 40 рублей.

Итого полная разработка веб-сервиса составляет 20644 рублей.

6.3 Методика обоснования цены

В современных рыночных экономических условиях веб-приложение выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид веб-приложения, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки веб-приложения являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования веб-приложения. Выбор эффективных проектов веб-приложения связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации веб-приложения, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

- снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
- сокращения расходов на оплату машинного времени и других ресурсов на отладку программ;
- снижения расходов на материалы;
- ускорение ввода в эксплуатацию новых систем;
- улучшения показателей основной деятельности в результате использования веб-приложения.

Стоимостная оценка веб-приложения у разработчиков предполагает определение затрат, что включает следующие статьи:

- заработная плата исполнителей – основная и дополнительная;
- отчисления в фонд социальной защиты населения;
- отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
- расходы на материалы и комплектующие;
- расходы на спецоборудование;
- расходы на оплату машинного времени;
- прочие прямые затраты;
- накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена веб-приложения.

6.3.1 Объем веб-приложения

Для оценки объема веб-приложения, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем веб-приложения V_o , вычисляется как сумма объемов V_i каждой из n его функций (формула 6.1).

$$V_o = \sum_{i=1}^n V_i. \quad (6.1)$$

В таблице 6.2 представлены функции присутствующие в рассматриваемом программном средстве и соответствующий им объем в условных машино-командах [33].

Таблица 6.2 – Содержание и объем функций в программном средстве

| № функции | Содержание функции | Объем, условных машино-команд |
|-----------|--|-------------------------------|
| 101 | Организация ввода информации | 130 |
| 102 | Контроль, предварительная обработка и ввод информации | 490 |
| 107 | Организация ввода/вывода информации в интерактивном режиме | 280 |
| 109 | Управление вводом/выводом | 1970 |
| 203 | Обработка наборов и записей базы данных | 2370 |
| 206 | Манипулирование данными | 7860 |
| 207 | Организация поиска и поиск в базе данных | 4720 |
| 305 | Формирование файла | 2130 |
| 506 | Обработка ошибочных и сбойных ситуаций | 1540 |
| 507 | Обеспечение интерфейса между компонентами | 1680 |
| 604 | Справка и обучение | 720 |
| 707 | Графический вывод результатов | 420 |
| 708 | Интерактивный редактор текста | 3780 |
| 801 | Простой поиск контента портала | 55 |
| 806 | Сбор статистики о посетителях сайта | 95 |
| 810 | Формирование базы данных портала | 1480 |
| 811 | Администрирование и обновление сайта | 90 |
| | Итого | 29810 |

Опираясь на данные таблицы 6.2, можно определить объем веб-приложения разработанного в ходе дипломного проектирования:

$$V_o = 130 + 490 + 280 + 1970 + 2370 + 7860 + 4720 + 2130 + 1540 + 1680 + 720 + 420 + 3780 + 55 + 95 + 1480 + 90 = 29810 \text{ (условных машино-команд)}.$$

Уточненный объем веб-приложения V_o' равен произведению объема веб-приложения V_o на коэффициент изменения скорости обработки информации $K_{ск}$ (формула 6.2).

$$V_o' = V_o \cdot K_{ск} \quad (6.2)$$

где V_o' – уточненный объем веб-приложения, условных машино-команд;
 V_o – общий объем веб-приложения, условных машино-команд;
 $K_{ск}$ – коэффициент изменения скорости обработки информации.

Коэффициент изменения скорости обработки информации $K_{ск}$ равен 0,6; $V_o = 13687$ условных машино-команд – подсчитано по формуле (6.1).

Исходя из вычисленного объема веб-приложения, можно определить уточненный объем веб-приложения:

$$V_o' = 29810 \cdot 0,6 = 17886 \text{ (условных машино-команд).}$$

6.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере веб-программирования на Java. В итоге было установлено, что средняя месячная заработная плата на позиции junior/middle составляет 1500 рублей.

Проект разрабатывался одним человеком на протяжении четырех месяцев. Таким образом, основная заработная плата будет рассчитываться по формуле (6.3):

$$C_{оз} = T_{раз} \cdot K_{раз} \cdot C_{зп}, \quad (6.3)$$

где $C_{оз}$ – основная заработная плата, руб.;
 $T_{раз}$ – время разработки, месяцев;
 $K_{раз}$ – количество разработчиков, человек;
 $C_{зп}$ – средняя месячная заработная плата, руб.

$$C_{оз} = 3 \cdot 1 \cdot 1500 = 4500 \text{ (руб).}$$

6.3.3 Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (6.4):

$$C_{дз} = \frac{C_{оз} \cdot H_{дз}}{100}, \quad (6.4)$$

где $C_{оз}$ – основная заработная плата, руб.;
 $H_{дз}$ – норматив дополнительной заработной платы, %.

$$C_{дз} = 4500 \cdot 10 / 100 = 450 \text{ (руб).}$$

6.3.4 Отчисления в Фонд социальной защиты населения и Белгосстрах

Отчисления в Фонд социальной защиты населения и Белгосстрах (ФСЗН) определяются в соответствии с действующими законодательными актами по

нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле (6.5):

$$C_{\text{фсзн}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot H_{\text{фсзн}}}{100}, \quad (6.5)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработная плата за веб-приложение, млн. руб.;

$H_{\text{фсзн}}$ – норматив отчислений в Фонд социальной защиты населения, %.

$$C_{\text{фсзн}} = (4500 + 450) \cdot 34,4 / 100 = 1702,8 \text{ (руб.)}.$$

6.3.5 Расходы на материалы

Сумма расходов на материалы C_M определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода H_M на уточненный объем веб-приложения V_o' (формула 6.6).

$$C_M = H_M \cdot \frac{V_o'}{100}. \quad (6.6)$$

где C_M – сумма расходов на материалы, руб.;

H_M – норма расхода материалов в расчете на 100 строк исходного кода веб-приложения, руб.;

V_o' – уточненный объем веб-приложения, условных машино-команд.

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равен 0,046 руб (по данным, приведенным в приложении 2 таблице П 2.10 «Оценка значений среднего расхода материалов на разработку и отладку 100 строк кода применения веб-приложения» методического пособия [33]), можно определить сумму расходов на материалы:

$$C_M = 0,0460 \cdot 17886 / 100 = 8,23 \text{ (руб.)}.$$

Таким образом, расходы на материалы составляют 8,23 рублей

6.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени $C_{\text{мв}}$ определяется как произведение стоимости одного машино-часа $C_{\text{мч}}$ на уточненный объем веб-приложения V_o' и на норматив расхода машинного времени на отладку ста строк исходного кода $H_{\text{мв}}$ (формула 6.7).

$$C_{\text{мв}} = C_{\text{мч}} \cdot \frac{V_o'}{100} \cdot H_{\text{мв}}. \quad (6.7)$$

где $C_{\text{мв}}$ – сумма расходов на оплату машинного времени, руб.;

$C_{\text{мч}}$ – цена одного машино-часа, руб.;

$H_{\text{мв}}$ – норматив расхода машинного времени на отладку 100 строк программного кода, машино-часов;

V_o' – уточненный объем веб-приложения, условных машино-команд.

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 15 (по данным, приведенным в приложении 2 таблице П 2.11 «Оценка значений среднего машинного времени на отладку 100 строк исходного кода без применения веб-приложения» методического пособия [33]), можно определить сумму расходов на оплату машинного времени:

$$C_{\text{мв}} = 0,05 \cdot 17886 \cdot 10 / 100 = 89,43 \text{ (руб.)}.$$

Таким образом, расходы на оплату машинного времени составляют 89,43 рублей.

6.3.7 Прочие прямые затраты

Сумма прочих затрат $C_{\text{пз}}$ определяется как произведение основной заработной платы исполнителей на конкретное программное средство $C_{\text{оз}}$ на норматив прочих затрат в целом по организации $H_{\text{пз}}$ (формула 6.8).

$$C_{\text{пз}} = \frac{C_{\text{оз}} \cdot H_{\text{пз}}}{100}. \quad (6.8)$$

$$C_{\text{пз}} = 4500 \cdot 10 / 100 = 450 \text{ (руб.)}.$$

Таким образом, прочие прямые затраты составили 450 рублей.

где $C_{\text{пз}}$ – сумма прочих затрат, руб.;

$H_{\text{пз}}$ – норматив прочих затрат в целом по организации, %;

$C_{\text{оз}}$ – основная заработная плата, руб.

6.3.8 Накладные расходы

Сумма накладных расходов $C_{\text{обп, обх}}$ – произведение основной заработной платы исполнителей на конкретное программное средство $C_{\text{оз}}$ на норматив накладных расходов в целом по организации $H_{\text{обп, обх}}$ (формула 6.9).

$$C_{\text{обп, обх}} = \frac{C_{\text{оз}} \cdot H_{\text{обп, обх}}}{100}. \quad (6.9)$$

где $C_{\text{нр}}$ – сумма накладных расходов, руб.;

$H_{\text{нр}}$ – норматив накладных расходов в целом по организации, %;

$C_{\text{пз}}$ – сумма прочих затрат, руб.;

$C_{\text{оз}}$ – основная заработная плата, руб.

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

$$C_{\text{обп, обх}} = 4500 \cdot 100 / 100 = 4500 \text{ (руб.)}.$$

6.3.9 Сумма расходов на разработку веб-приложения

Сумма расходов на разработку веб-приложения $C_{\text{р}}$ определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное

программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов (формула 6.10).

$$C_p = C_{оз} + C_{дз} + C_{фсзн} + C_m + C_{мв} + C_{пз} + C_{обп, обх} \quad (6.10)$$

где C_p – сумма расходов на разработку веб-приложения, руб.;

$C_{оз}$ – основная заработная плата, руб.;

$C_{дз}$ – дополнительная заработная плата на конкретное веб-приложения, руб.;

$C_{фсзн}$ – сумма отчислений в Фонд социальной защиты населения, руб.;

C_m – сумма расходов на материалы, руб.;

$C_{мв}$ – сумма расходов на оплату машинного времени, руб.;

$C_{пз}$ – сумма прочих затрат, руб.;

$C_{обп, обх}$ – сумма накладных расходов, руб.

$$C_p = 4500 + 450 + 1702,8 + 8,23 + 89,43 + 450 + 4500 = 11700,46 \text{ (руб.)}$$

Таким образом, сумма расходов на разработку веб-приложения составила 11700,46 рублей.

6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию веб-приложения $C_{рса}$ определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию $H_{рса}$ (формула 6.11).

$$C_{рса} = \frac{C_p \cdot H_{рса}}{100} \quad (6.11)$$

$$C_{рса} = 11700,46 \cdot 10 / 100 = 1170 \text{ (руб.)}$$

где $C_{рса}$ – сумма расходов на сопровождение и адаптацию веб-приложения, руб.;

C_p – сумма расходов на разработку веб-приложения, руб.;

$H_{рса}$ – норматив расходов на сопровождение и адаптацию, %.

Таким образом, сумма расходов на сопровождение и адаптацию веб-приложения составляют 1170 рублей.

6.3.11 Полная себестоимость

Полная себестоимость C_n определяется как сумма двух элементов: суммы расходов на разработку C_p и суммы расходов на сопровождение и адаптацию веб-приложения $C_{рса}$ (формула 6.12).

$$C_n = C_p + C_{рса} \quad (6.12)$$

где C_n – полная себестоимость веб-приложения, руб.;

C_p – сумма расходов на разработку веб-приложения, руб.;

$C_{рса}$ – сумма расходов на сопровождение и адаптацию веб-приложения, руб.

$$C_{\pi} = 11700,46 + 1168,1 = 12868,56 \text{ руб.}$$

Полная себестоимость составила 12868,5 рублей.

6.3.12 Определение цены, оценка эффективности

Исходя из рыночной стоимости аналогичных продуктов, которую получили при маркетинговом анализе, можно рассчитать прибыль и рентабельность проекта.

Цена разработчика программного продукта C_p определяется разностью среднерыночной цены C_{cp} и НДС (формулы 6.13).

$$C_p = C_{cp} - \text{НДС}, \quad (6.13)$$

$C_{cp} = 20644$ руб. – получена в результате маркетингового анализа; $\text{НДС} = 20\%$.

$$C_p = 20644 / 1,2 = 17203,3 \text{ руб.}$$

Прибыль P_p определяется как разность цены разработчика C_p и полной себестоимости C_{π} . (формула 6.14).

$$P_p = C_p - C_{\pi} \quad (6.14)$$

$$P_p = 17203,3 - 12868,56 = 4334,74 \text{ руб.}$$

Рентабельность продукта R_{π} определяется как отношение прибыли P_p к себестоимости C_{π} (формула 6.15).

$$R_{\pi} = \frac{P_p}{C_{\pi}} \cdot 100\% \quad (6.15)$$

$$R_{\pi} = (4334,74 / 12868,56) \cdot 100 = 33,69\%$$

Ожидаемый эффект показывает, как и насколько влияет разработанный продукт. В данном случае ожидаемый эффект – это увеличение пользователей интернет площадки для проведения торгов.

Вывод по разделу 6

В таблице 6.3 представлены результаты расчетов для основных показателей данной главы в краткой форме.

Таблица 6.3 – Результаты расчетов

| Наименование показателя | Значение |
|---|----------|
| Время разработки, мес. | 3 |
| Количество программистов, чел. | 1 |
| Зарплата с отчислениями, руб. | 6652,8 |
| Расходы на материалы, оплату машинного времени, прочие, руб | 547,66 |
| Накладные расходы, руб | 4500 |
| Себестоимость разработки веб-приложения, руб | 11700,46 |
| Расходы на сопровождение и адаптацию, руб. | 1170 |
| Полная себестоимость, руб. | 12868,56 |
| Цена аналога, руб. | 20644 |
| Прибыль от реализации, руб. | 4334,74 |
| Рентабельность разработки, % | 33,69 |

Разработка программного средства одним работником в течении 3-х месяцев при заданных условиях обойдется в 11700,46 рублей. Реализация указанного приложения по среднерыночной цене 20644 рублей принесет прибыль организации в сумме 4334,74 рубля, уровень рентабельности составит 33,69%.

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проектирования было разработано веб-приложение для проведения аукционных торгов.

Ключивые возможности и особенности разработанного веб-приложения «Аукцион» для пользователя:

- регистрация пользователя;
- авторизация пользователя;
- изменения данных в профиле пользователя;
- удаление профиля пользователя;
- управление лотами;
- участие в торгах;
- скачивание документа с историй ставок;
- получение документа подтверждающего состоявшиеся торги;
- API.

Для администратора были реализованы следующие возможности:

- панель администратора;
- управление лотами;
- управление пользователями;
- просмотр статистики в графиках;
- просмотр статистики в числах;
- получение документа с отчетом на почту или прямое скачивание с сервера.

В разделе анализа предметной области были рассмотрены аналоги разрабатываемого веб-приложения, определены плюсы и минусы существующих аналогов.

В разделе проектирования веб-приложения, был обоснован выбор технологий, которые применены в разработке дипломного проекта. Серверная часть реализована с использованием объектно-ориентированного языка программирования Java. Для отображения графической части реализовано с применением JSP страниц, HTML5, JavaScript и CSS3. Тестирование проведено вручную и с использованием unit-тестов, результат работы unit-тестов отображен с использованием плангина JaCoCo. Система контроля версий – GitHub. Сервер – Apache Tomcat.

Веб-приложение является законченным программным продуктом. Использование веб-приложения не составит труда для пользователей любого возраста, пола, расы, так как имеет удобный и интуитивно понятный интерфейс.

Данное приложение соответствует поставленным задачам и отвечает всем требованиям, необходимым для пользования данным программным продуктом.

Работа промежуточного варианта приложения демонстрировалась на 69 научно-технической конференции учащихся, студентов и магистрантов в Белорусском государственном технологическом университете 21 апреля 2018 года.

| | | | | | | | | | |
|-------------|------|---------------|---------|------|---------------------|------|--------|--|--|
| | | | | | ДП 10.00. ПЗ | | | | |
| | | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | | | | | |
| Разраб. | | Жигало В.Ю. | | | Заключение | | | | |
| Провер. | | Наркевич А.С. | | | | | | | |
| Консультант | | Наркевич А.С. | | | | | | | |
| Н. Контр. | | Жиляк Н.А. | | | | | | | |
| Утверд. | | Пацей Н.В. | | | | | | | |
| | | | | | Лит. | Лист | Листов | | |
| | | | | | у | 1 | 1 | | |
| | | | | | БГТУ 74122013, 2018 | | | | |

Список использованных источников

- 1 Аукцион [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%90%D1%83%D0%BA%D1%86%D0%B8%D0%BE%D0%BD/>. Дата доступа: 21.03.2017.
- 2 Аукционы Беларуси — Ау.by [Электронный ресурс] – Режим доступа: <http://au.by/>. Дата доступа: 21.03.2017.
- 3 XLOT – крупнейший интернет-аукцион в Беларуси [Электронный ресурс] – Режим доступа: <http://xlot.by/>. Дата доступа: 21.03.2017.
- 4 БелАукцион - интернет-аукцион Беларуси [Электронный ресурс] – Режим доступа: <http://belauction.by/>. Дата доступа: 21.03.2017.
- 5 Проектирование программного обеспечения [Электронный ресурс] – Режим доступа: goo.gl/u2UFSn. Дата доступа: 21.03.2017.
- 6 Архитектура программного обеспечения [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Архитектура_программного_обеспечения. Дата доступа: 21.03.2017.
- 7 UML [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/UML>. Дата доступа: 21.03.2017.
- 8 Блок-схема [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA-%D1%81%D1%85%D0%B5%D0%BC%D0%B0>. Дата доступа: 21.03.2017.
- 9 Сравнение MySQL и PostgreSQL [Электронный ресурс] – Режим доступа: <https://hyperhost.ua/info/sravnenie-mysql-i-postgresql/>. Дата доступа: 21.03.2017.
- 10 Основы Hibernate [Электронный ресурс] – Режим доступа: <https://habr.com/post/29694/>. Дата доступа: 30.03.2017.
- 11 ORM [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/ORM>. Дата доступа: 30.03.2017.
- 12 Apache Tomcat [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Apache_Tomcat. Дата доступа: 30.03.2017.
- 13 Apache Maven [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Apache_Maven. Дата доступа: 30.03.2017.
- 14 Apache Maven — основы [Электронный ресурс] – Режим доступа: <https://habr.com/post/77382/>. Дата доступа: 30.03.2017.
- 15 JavaServer Pages [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/JavaServer_Pages. Дата доступа: 30.03.2017.
- 16 WEB: что такое JSP [Электронный ресурс] – Режим доступа: java-course.ru/student/book1/jsp/. Дата доступа: 30.03.2017.
- 17 SHA-2 [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/SHA-2>. Дата доступа: 04.04.2017.
- 18 Обзор Gson - работаем с JSON в Java [Электронный ресурс] – Режим доступа: www.javenue.info/post/gson-json-api. Дата доступа: 04.05.2017.

| | | | | | | | | |
|-------------|-------------|-----------------|----------------|-------------|---|----------------------------|-------------|---------------|
| | | | | | <i>ДП 11.00. ПЗ</i> | | | |
| <i>Изм.</i> | <i>Лист</i> | <i>№ докум.</i> | <i>Подпись</i> | <i>Дата</i> | | | | |
| Разраб. | | Жигало В.Ю. | | | <i>Список использованных источников</i> | <i>Лит.</i> | <i>Лист</i> | <i>Листов</i> |
| Провер. | | Наркевич А.С. | | | | У | 1 | 2 |
| Консультант | | Наркевич А.С. | | | | <i>БГТУ 74122013, 2018</i> | | |
| Н. Контр. | | Жуляк Н.А. | | | | | | |
| Утверд. | | Пацей Н.В. | | | | | | |

19 Измерение покрытия java-бекенда [Электронный ресурс] – Режим доступа: artkoshelev.github.io/posts/runtime-coverage. Дата доступа: 04.05.2017.

20 Вышел FindBugs 3.0.1 [Электронный ресурс] – Режим доступа: <https://habr.com/post/251749/>. Дата доступа: 04.05.2017.

ПРИЛОЖЕНИЕ А. Листинг исходного кода генерации отчета истории ставок на лот PDF

```

public void generateDocHistoryBetPDF(String uuidLot, HttpServletRequest req, HttpServletResponse
resp, boolean isSendMail) {
    try {
        Document document = new Document(PageSize.A4);
        URL url = new URL(req.getRequestURL().toString());
        urlLot = url.getProtocol() + "://" + url.getHost() + ":" + url.getPort() + RESIRECT_URL_PATH
+ uuidLot;
        documentPasscode = valueOf(UUID.randomUUID()).substring(0, 8);
        PdfPTable table = new PdfPTable(new float[]{20, 10, 10, 10});
        table.setTotalWidth(PageSize.A4.getWidth() - 10);
        table.setLockedWidth(true);
        table.getDefaultCell().setHorizontalAlignment(Element.ALIGN_LEFT);
        table.addCell("First/Last name");
        table.addCell("Bet");
        table.addCell("Date");
        table.addCell("Time");
        table.setHeaderRows(1);
        PdfPCell[] cells = table.getRow(0).getCells();
        for (PdfPCell cell : cells) {
            cell.setBackgroundColor(BaseColor.WHITE);
        }
        List<BetHistoryTO> list = getHistoryBets(uuidLot);
        for (int i = 1; i < list.size(); i++) {
            table.addCell(list.get(i).getUserName());
            table.addCell(valueOf(list.get(i).getBet()));
            table.addCell(list.get(i).getDate());
            table.addCell(list.get(i).getTime());
        }
        PdfWriter pdfWriter;
        // check api call
        if (!StringUtils.isBlank(req.getParameter(PARAMETER_API_KEY_NAME)) && !isSendMail)
        {
            resp.setContentType("application/json");
            pdfWriter = PdfWriter.getInstance(document, byteArrayOutputStreamPDF);

        } else {
            if (!isSendMail) {
                resp.setHeader("Content-Disposition", "attachment;filename=" + fileName);
                resp.setContentType("application/pdf;charset=UTF-8");
                pdfWriter = PdfWriter.getInstance(document, resp.getOutputStream());
            } else {
                pdfWriter = PdfWriter.getInstance(document, byteArrayOutputStreamPDF);
            }
        }
        // passcode for document
        pdfWriter.setEncryption(
            getDocumentPasscode().getBytes(StandardCharsets.UTF_8),
            PDF_OWNER_PASSCODE.getBytes(StandardCharsets.UTF_8),
            PdfWriter.ALLOW_COPY,
            PdfWriter.ENCRYPTION_AES_128
    
```

```

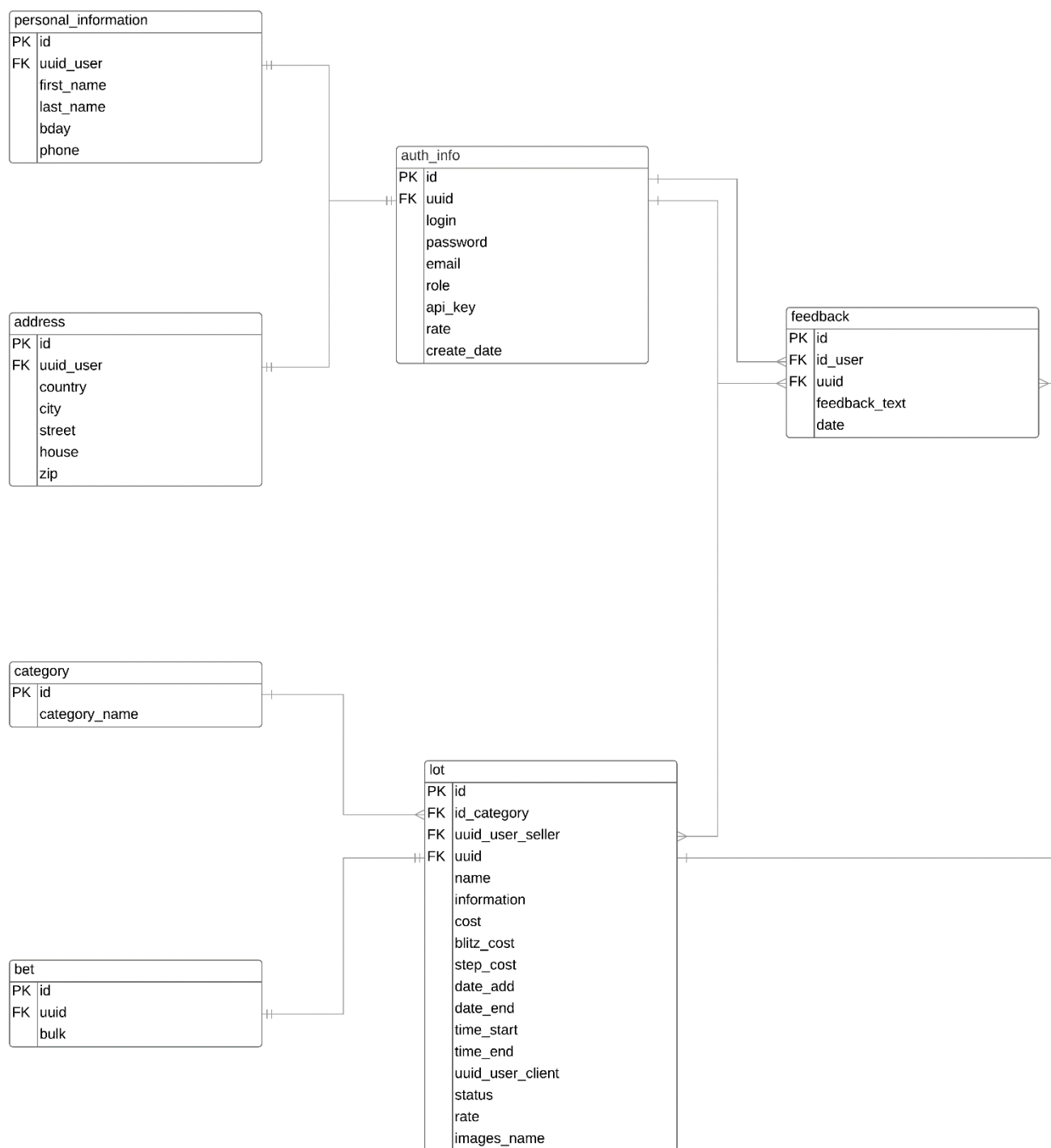
);

document.open();
document.addAuthor("Auction Diploma");
document.addTitle(TITLE_BET_HISTORY);
document.add(new Paragraph("Auction Diploma"));
document.add(new Paragraph(TITLE_BET_HISTORY));
document.add(new Paragraph(valueOf(new
SimpleDateFormat(PATTERN_FULL_DATE_TIME).format(new Date()))));

BarcodeQRCode barcodeQRCode = new BarcodeQRCode(getLotUrl(), 1000, 1000, null);
Image codeQrImage = barcodeQRCode.getImage();
codeQrImage.setAbsolutePosition(469, 729);
codeQrImage.scaleAbsolute(100, 100);
document.add(codeQrImage);
document.add(new Paragraph("\n-----" +
    "-----"));
document.add(new Paragraph("\n"));
document.add(new Paragraph("Rate: " + getRate(uuidLot, LOT)));
document.add(new Paragraph("\n"));
if (list.size() != 1)
    document.add(table);
else
    document.add(new Paragraph("Bets isn't found"));
document.add(new Paragraph("\n"));
document.add(new Paragraph("-----" +
    "-----"));
document.add(new Paragraph("UUID lot: " + uuidLot));
document.add(new Paragraph("URL lot: " + getLotUrl()));
document.close();
if (isSendMail) {
    MailUtil mailUtil = new MailUtil();
    mailUtil.addAttachment(prepareFileForAttach(byteArrayOutputStreamPDF,
        fileName, PDF_EXTENSION)
    );
    String body = "<br/>" + timeNow +
        "<br/>Добрый день, найдите прикрепленные файлы в письме." +
        "<br/>Адрес лота: <b>" + urlLot + "</b>" +
        "<br/>Пароль для открытия файла: <b>" + documentPasscode + "</b>" +
        "<br/><br/>С уважением";
    mailUtil.sendSimpleHtmlMail(getUserEmailByUUID(getUUIDUserByUUIDLot(uuidLot)), body,
    subject + uuidLot);
    } else {
        LOGGER.info("Send document not required");
    }
    LOGGER.info("PDF document successfully generated");
    LOGGER.info("Document name\t" + fileName);
    LOGGER.info("Password\t" + getDocumentPasscode());
} catch (IOException | DocumentException e) {
    new MailUtil().sendErrorMail(Arrays.toString(e.getStackTrace()));
    LOGGER.error(e.getLocalizedMessage());
}
}

```

ПРИЛОЖЕНИЕ Б. Логическая схема базы данных



ПРИЛОЖЕНИЕ В. Блок-схема алгоритма регистрации и/или авторизации пользователя

