

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА БАКАЛАВРА

на тему

Система онлайн торговли «бизнес-потребитель»

Студент

27.05.20 Тимирязцев Андрей Владимирович
(подпись, дата) (ФИО полностью)

Руководитель

27.05.20 С.А. Зинкин
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

27.05.20 А.В. Кучин
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой

27.06.2020 М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от
17.06.2020 № 15)

Секретарь ГЭК

Е. Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

« 30 » апреля 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Тимирязев Андрей Владимирович гр. 16ВВ1 факультета вычислительной техники.

2. Тема работы: «Система онлайн торговли «бизнес - потребитель».

Тема утверждена приказом ПГУ №0335а/о от "30" апреля 2020 г

3. Руководитель работы д.т.н., профессор кафедры ВТ Зинкин С.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Назначение разработки – Система предназначена для разработки клиентского серверного приложения для приема и обработки онлайн товарооборота.

5. Перечень подлежащих разработке вопросов.

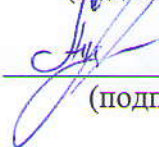
В процессе работы определен список требований программного обеспечения, разработаны функции, позволяющие приложению вести обработку товаров, каталогов товаров. В системе предусмотрена работы с системой платежей и авторизации.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	30.04.2020-02.05.2020	
2.	Выбор архитектуры приложения	10	03.05.2020-05.05.2020	
3.	Составление требований к системе	20	06.05.2020-16.05.2020	
4.	Реализация приложения	10	17.05.2020-23.05.2020	
5.	Отладка системы	10	24.05.2020-27.05.2020	
6.	Моделирование и поиск уязвимостей, исправление ошибок в работе системы	10	28.05.2020-30.05.2020	
7.	Оформление выпускной работы	30	31.05.2020-03.06.2020	

Дата выдачи задания "30" апреля 2020 г.

Руководитель бакалаврской работы  30.04.2020 Зинкин С.А.
(подпись, дата)

Задание к исполнению принял студент  30.04.2020 Тимирязев А.В.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.

РЕФЕРАТ

Дипломная работа содержит 114 листов, 48 рисунков, 3 приложения.

WEB ПРИЛОЖЕНИЕ, БАЗА ДАННЫХ , ТРЕХЗВЕННАЯ АРХИТЕКТУРА.

Цель работы – разработать масштабируемую, отказоустойчивую систему продажи товаров на базе паттерна модель, вид, контроллер.

Объектом разработки является программное приложение, обеспечивающее клиентам доступ к каталогу товаров и услуг, администраторам доступ к настройкам системы на базе трехзвенной архитектуры.

В результате проведенной работы должен быть разработан проект, удовлетворяющий поставленной задаче, а именно, создан программный комплекс – «тонкий» клиент, серверная бизнес логика и СУБД, и разработаны соответствующие алгоритмы, реализующие возможность хранить, обрабатывать и доставлять до потребителя информацию в виде сообщений.

Данное приложение рекомендуется использовать для массового продвижения товаров и услуг. Система может быть внедрена в сегменте электронной коммерции компании сегмента масс-маркет.

					<i>ПГУ 2.090301.36.001 ПЗ</i>						
<i>Изм</i>	<i>Лист</i>	<i>№ докум</i>	<i>Подпись</i>	<i>Дата</i>	<i>Система онлайн торговли «бизнес-потребитель» Пояснительная записка</i>			<i>Литера</i>	<i>Лист</i>	<i>Листов</i>	
<i>Разраб</i>		<i>Тимирязев А.В</i>									
<i>Пров</i>		<i>Зинкин С.А</i>							2	114	
<i>Н. Контр.</i>		<i>Кучин А.В</i>						<i>ФВТ, каф. ВТ гр.16ВВ1</i>			
<i>Утв</i>		<i>Митрохин М.А</i>									

Содержание

Введение	5
1. ИССЛЕДОВАНИЕ	7
1.1 Анализ предметной области	7
1.2 Требования к задачам системы.....	7
1.3 Функционал приложения	9
1.4 Выбор архитектуры приложения	9
1.4.1 Паттерн Цепочка	10
1.4.2 Паттерн Наблюдатель.....	11
1.4.3 Паттерн Шаблонный метод	11
1.4.4 Паттерн MVC.....	12
1.5 Выбор СУБД для хранения информации	13
1.6 Выбор инструментария разработки	14
1.6.1 Выбор языка программирования.....	15
1.6.2 Выбор вспомогательных элементов разработки	22
1.7 Выбор платежной системы для реализации оплаты в разрабатываемой системе.....	23
1.7.1 Система Payline	24
1.7.2 Система Shopify.....	24
1.7.3 Система TransferWise	25
1.7.4 Система Square	26
1.8 Выводы по главе.....	26
2. Методы и средства решения задачи.....	28
2.1 Требования к функциональному назначению приложения.....	28
2.2 Проектирование системы	29
2.3.Проектирование структур данных	39
2.3.1 Концептуальное проектирование.....	39

2.3.2 Даталогическое проектирование	43
2.3.3 Инфологическая модель	53
2.4 Проектирование пользовательского интерфейса	56
2.6 Интеграция платежной системы в структуру приложения	62
2.6.1 Разовые Платежи.....	65
2.6.2 Обработка платежа	69
2.6.3 Валидация платежа	70
3. Технология разработки и отладки web-приложений в среде NetBeans	73
3.1 Создание проекта в NetBeans.....	73
3.2 Отладка кода с помощью XDebug.....	78
3.2.1 Как работает отладка	80
4. Безопасность жизнедеятельности. Эргономика рабочего места программиста.....	83
4.1 Характеристика условий труда программиста	83
4.2 Эргономические требования к рабочему месту программиста	84
4.3 Освещенность рабочего места программиста.....	90
4.4 Расчет освещения рабочего места программиста.....	92
ЗАКЛЮЧЕНИЕ	95
ПРИЛОЖЕНИЕ А.....	96
ПРИЛОЖЕНИЕ Б.....	106
ПРИЛОЖЕНИЕ В.....	109

ВВЕДЕНИЕ

В современном мире электронная торговля занимает одно из ведущих мест ведения бизнеса: она обеспечивает необходимыми товарами и услугами, объединяет континенты и страны.

Благодаря онлайн торговле, товар может быть доставлен в отдаленные уголки мира. В настоящее время термин «электронная коммерция» трактуется по-разному. Это связано с тем, что данный вид торговли появился относительно недавно, а с другой стороны, ученые-экономисты с разных аспектов рассматривают сущность данного понятия. Например, российские экономисты определяют электронную коммерцию как «сферу экономики, которая включает в себя все финансовые и торговые транзакции, осуществляемые при помощи компьютерных сетей, и бизнес-процессы, связанные с проведением таких транзакций».

Большое число людей, особенно молодежь, очень часто предпочитает покупку товаров в Интернете, нежели в традиционных магазинах. Прежде всего, это объясняется приемлемыми ценами, а также более широким ассортиментом товара. Помимо этого затрата времени на совершение покупки посредством интернет-технологий гораздо меньше.

На сегодняшний день функционируют и развиваются такие отрасли предпринимательства, которые наиболее соответствуют интересам и потребностям населения. Данный факт можно рассмотреть на примере Amazon.com - крупнейшей американской компании, занимающейся в сфере онлайн торговли.

Продажа книг на начальном этапе работы компании со временем переросла в более широкий ассортимент, куда входили видеоигры, медикаменты, автомобили и т. д.

К 2019 г. товарооборот данной компании превысил 61 млрд долларов.

Несмотря на ряд преимуществ совершения онлайн покупок, существуют моменты онлайн системы требующие проработки: разработка

					ПГУ 2.090301.36.001 ПЗ	Лист
						5
Изм.	Лист	№ докум	Подпись	Дата		

интуитивно понятного интерфейса, разработка защищенной системы оплаты, разработка поисковой и рекомендательных систем внутри системы, разработки системы поиска контентной информации в интернете.

					ПГУ 2.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		6

1. ИССЛЕДОВАНИЕ

1.1 Анализ предметной области

Создание сайта электронной коммерции требует проработки архитектуры и информационных потоков приложения. Если разработка сайт идет поэтапно (agile подход), можно уменьшить первоначальные инвестиции и, следовательно, сократить свои потери, если идея окажется неудачной.

Данный подход позволяет использовать результаты ранней фазы, для оценки, стоит ли добавлять дополнительные функции.

Исходя из выше сказанного принято решение, что в первой главе требуется описать функционал приложения, выбрать архитектурное решение приложения и определиться со стеком технологий для реализации проекта.

При этом анализ и разработка приложения будет учитывать при реализации следующие риски: взлом электронного ресурса и СУБД, хаккинг банковской информации, аппаратные сбои на стороне хостинга, ненадежность системы доставки товара, ошибки программного обеспечения и изменение законодательства.

1.2 Требования к задачам системы

Система должна соответствовать следующим требованиям при работе администраторов, клиентов и гостей приложения.

Таблица 1 - Задачи решаемые разрабатываемой системой

№	Описание минимальных требований
1	На главной странице приложения должны отображаться доступные продукты, количество которых не меньше одного.
2	Клиенты должны иметь возможность добавлять товары в корзину.
3	Клиенты должны иметь возможность просматривать товары в корзине.
4	Клиенты должны иметь возможность обновлять количество товара в корзине.
5	Клиенты должны иметь возможность удалить любой продукт из корзины.

Продолжение Таблицы 1

№	Описание минимальных требований
6	Клиент должен иметь возможность опустошить все товары в корзине.
7	Клиенты должны иметь возможность подписаться на список рассылки.
8	Клиенты должны иметь возможность оформить заказ через платежное решение PayPal Express Checkout в Paypal Sandbox.
9	Клиенты должны иметь возможность просматривать подтверждение заказа после успешного завершения заказа.
10	Администратор должен пройти аутентификацию, чтобы иметь доступ к странице администратора приложения для выполнения любой административной задачи.
11	Администратор должен иметь возможность управлять (добавлять, обновлять и удалять) продуктами и их категориями.
12	Администратор должен иметь возможность просматривать списки товаров и категорий.
13	Администратор должен иметь возможность просматривать список подписчиков электронной почты.
14	Администратор должен иметь возможность просматривать информацию об оплате и заказе.
15	Администратор должен иметь возможность обновлять статус заказа.
16	Приложение должно сохранить все данные о клиенте, продукте, заказе, оплате и администрировании в базе данных PostgreSQL.
Описание дополнительных требований	
17	Приложение должно отображать дружественное сообщение об ошибке после неудачного входа администратора.
18	Приложение должно отображать дружественное сообщение об ошибке после неудачного или отмененного процесса оплаты.
19	Приложение должно реализовывать интерактивные интерфейсы пользователя (GUI).
20	Подтверждение по электронной почте должно быть отправлено клиентам, которые подписались на список рассылки.
21	Подтверждение по электронной почте должно быть отправлено клиентам после успешного завершения заказа.
22	В приложении должны быть реализованы другие способы оплаты, такие как Google Pay, Yandex Card.

1.3 Функционал приложения

В процессе выполнения работы планируется проработать следующие вопросы:

- Разработать СУБД для хранения каталога товаров, категории продукты и иную информацию.
- Разработать структурированные запросы (SQL) и программный код (в рамках выбранного инструментария) для доступа к данным и сделать бизнес процессы ресурса функциональными.
- Добавить данные в каталог продуктов, которые определяют атрибуты продукта.
- Обеспечить поисковую систему в рамках электронного ресурса.
- Внедрить способы расширения поискового функционала.
- Внедрить роутинг URL и обработку ошибок перенаправления запросов пользователя.
- Разработать систему платежей в онлайн системе платежей (например, PayPal).
- Разработать для администраторов сайта личный раздел сайта, где они могут управлять каталогами товаров.
- Реализовать функцию «Корзина покупателя».
- Создать страницу администрирования заказа.
- Внедрить систему рекомендаций по продукту.

1.4 Выбор архитектуры приложения

При разработке программного обеспечения необходимо тщательно продумывать архитектуру, разбивая ее на отдельные более простые блоки и подзадачи. У разработчика первым делом возникают вопросы, как сделать решение проще и каким образом данный функционал был реализован ранее.

Дело в том, что почти наверняка подобные задачи уже решались ранее и уже существуют хорошо продуманные элегантные решения, составленные экспертами. Часто при разработке программного обеспечения используются

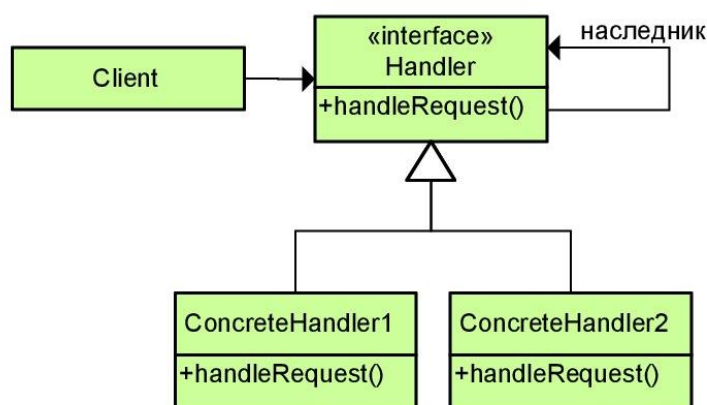
шаблоны проектирования или паттерны, которые представляют собой модель взаимодействия классов для решения какой-либо задачи.

Реализация проекта с использованием паттернов значительно упрощает процесс разработки. Основным плюсом шаблонов является наличие готовых абстрактных решений большинства задач. При этом каждый тип задач имеет свой шаблон с названием, что упрощает коммуникацию между разработчиками, повышает читаемость кода и понимание архитектуры проекта. С использованием шаблонов происходит унификация модулей и элементов проекта, значительно снижается количество ошибок.

Но нельзя забывать, что использование шаблонов проектирования без веской причины может усложнить архитектуру проекта и отрицательно сказаться на производительности. Ниже представлены паттерны, в рамках которых предполагается решение поставленной задачи.

1.4.1 Паттерн Цепочка

Цепочка обязанностей - поведенческий шаблон проектирования, предназначенный для организации в системе уровней ответственности.



Цепочка обязанностей *Chain of responsibility*

Тип: Поведенческий

Что это:

Избегает связывания отправителя запроса с его получателем, давая возможность обработать запрос более чем одному объекту. Связывает объекты-получатели и передаёт запрос по цепочке пока объект не обработает его.

Рисунок 1.1 – Паттерн Цепочка.

Например, есть три торговых отдела (А, В и С), настроенных на базе одной электронной площадки. На каждом лежит разное количество товара. На А есть 10 единиц, на В есть 30 единиц и на С - 5 единиц. Предпочтение

отдается в следующем порядке: А, В и С. Покупатель пытается заказать 45 единиц товара. Используя цепочку обязанностей, первым на возможность выдачи будет проверен метод А, и в случае успеха пройдет транзакция и цепь разорвется. Если нет, то запрос перейдет к методу В для аналогичной проверки. Здесь А, В и С - это звенья цепи, а все явление - цепочка обязанностей.

1.4.2 Паттерн Наблюдатель

Наблюдатель - поведенческий шаблон проектирования, также известен как «подчинённые» (Dependents). Создает механизм у класса, который позволяет получать экземпляру объекта этого класса оповещения от других объектов об изменении их состояния, тем самым наблюдая за ними.

Наблюдатель *Observer*

Тип: Поведенческий

Что это:

Определяет зависимость “один ко многим” между объектами так, что когда один объект меняет своё состояние, все зависимые объекты оповещаются и обновляются автоматически.

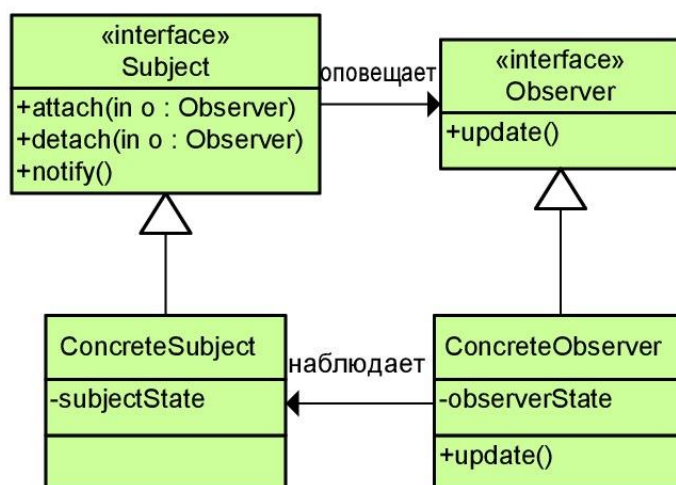


Рисунок 1.2 – Паттерн Наблюдатель.

Пример: клиенты, ищущие товары, подписываются на публикации на сайтах интеграторах и получают уведомления, когда появляются предложения подходящие по параметрам.

1.4.3 Паттерн Шаблонный метод

Шаблонный метод - поведенческий шаблон проектирования, определяющий основу алгоритма и позволяющий наследникам

переопределять некоторые шаги алгоритма, не изменяя его структуру в целом.

Шаблонный метод

Template method

Тип: Поведенческий

Что это:

Определяет алгоритм, некоторые этапы которого делегируются подклассам. Позволяет подклассам переопределить эти этапы, не меняя структуру алгоритма.

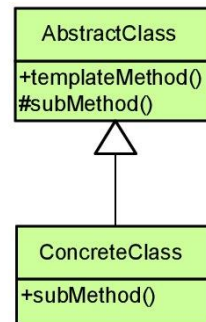


Рисунок 1.3 – Паттерн Шаблонный метод.

Пример требуется собрать сложный товар или конструкцию для продажи, например расчет строительства дома. Этапы будут такими: подготовка фундамента, возведение стен, настил крыши, настил перекрытий. Порядок этапов никогда не меняется. Нельзя настелить крышу до возведения стен и т. д. Но каждый этап модифицируется: стены, например, можно возвести из дерева, кирпича или бетона.

1.4.4 Паттерн MVC

Шаблон MVC описывает простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса. В результате, приложение легче масштабируется, тестируется, сопровождается и конечно же реализуется.

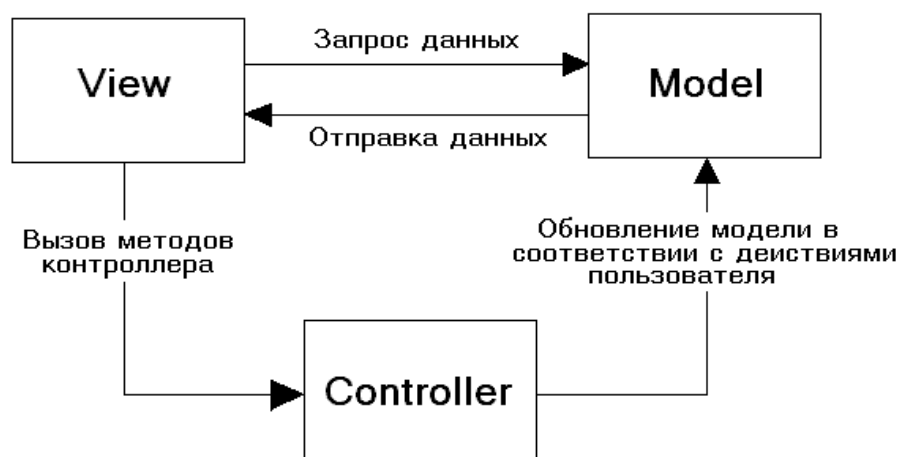


Рисунок 1.4 – Паттерн MVC.

Типичную последовательность работы MVC-приложения можно описать следующим образом:

При заходе пользователя на веб-ресурс, скрипт инициализации создает экземпляр приложения и запускает его на выполнение.

При этом отображается вид главной страницы сайта.

Приложение получает запрос от пользователя и определяет запрошенный контроллер и действие. В случае главной страницы, выполняется действие по умолчанию (index).

Приложение создает экземпляр контроллера и запускает метод действия, в котором, к примеру, содержатся вызовы модели, считывающие информацию из СУБД.

После этого, действие формирует представление с данными, полученными из модели и выводит результат пользователю.

Шаблон MVC используется в качестве архитектурной основы во многих фреймворках и CMS, которые создавались для того, чтобы иметь возможность разрабатывать качественные сложные решения за более короткий срок. Это стало возможным благодаря повышению уровня абстракции, поскольку есть предел сложности конструкций, которыми может оперировать человеческий мозг.

Веб-фреймворки типа Yii или Kohana успешно используют данное решение для разработки онлайн приложений. Именно поэтому было принято решение использовать данный паттерн как основу архитектуры разрабатываемого приложения.

1.5 Выбор СУБД для хранения информации

Выбор системы хранения данных определен скоростью и простотой разработки на свободно распространяемой платформе. Значит сравнение необходимо произвести между двумя продуктами MySQL и PostgreSQL.

Таблица 2 – сравнение MySQL и PostgreSQL

	MySQL	PostgreSQL
Ответственный за код	компания MySQL AB	разные разработчики
Сжатие данных при передаче	+	+
Поддержка внешних ключей	+	+
Репликация	+	частичная поддержка
Под запросы	+	+
Полнотекстовые индексы	+	+
Чистка после работы команд UPDATE и DELETE	не нуждается	принудительная
Система привилегий	+	частичная поддержка
Хранение таблиц в файлах	+	не поддерживает
Хранение/обработка географических данных	частичная поддержка	+
Лицензирование	GNU GPL	BSD

Для разработки интернет-магазина, при отсутствии в команде администратора (службы DevOps), до перерастания в большой проект при наличии опыта работы с MySQL - выбор MySQL рациональное решение. Если предполагаете, что проект будет популярным, если он большой, его будет тяжело переписать, если в нём сложная логика и связи между таблицами – PostgreSQL будет верным решением. В данном проекте выбор сделан в пользу первой СУБД.

1.6 Выбор инструментария разработки

Для реализации проекта требуется грамотно выбрать инструменты реализации вида, контроллера и модели.

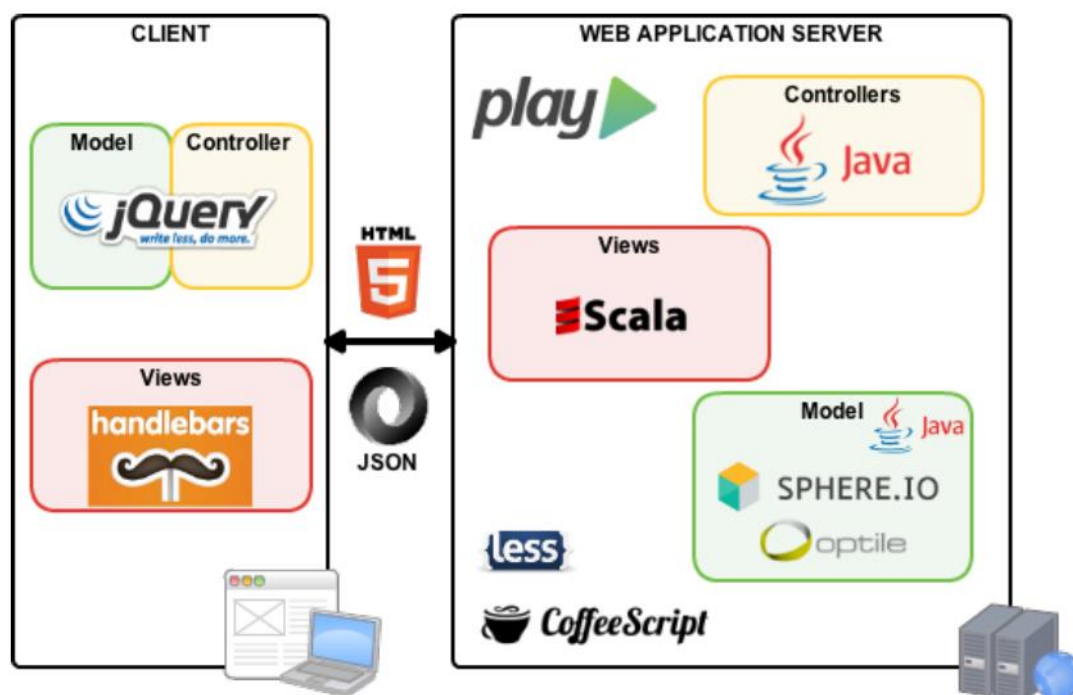


Рисунок 1.5 – Популярные инструменты реализации системы e-commerce.

Ниже представлены доводы выбора языка программирования, вспомогательных инструментов разработки и платежной системы для реализации системы e-commerce.

1.6.1 Выбор языка программирования

PHP - это интерпретируемый язык программирования общего назначения с открытым исходным кодом, является одним из лучших и востребованных в данной области. Основным преимуществом PHP сценариев является возможность без проблем интегрироваться в обычные html-документы, при этом сохраняя всю функциональность, конечно же, при условии наличия на веб-сервере интерпретатора языка PHP.

Главной областью применения PHP является написание программ сценариев, которые работают на стороне сервера. Помимо различных операций с содержимым html-документов, с помощью PHP можно форматировать HTTP-заголовки, устанавливать cookie, управлять аутентификацией, а также перенаправлять пользователя на другие страницы.

PHP предоставляет возможности по доступу к базам данных, позволяет проводить интеграцию с различными внешними библиотеками [1].

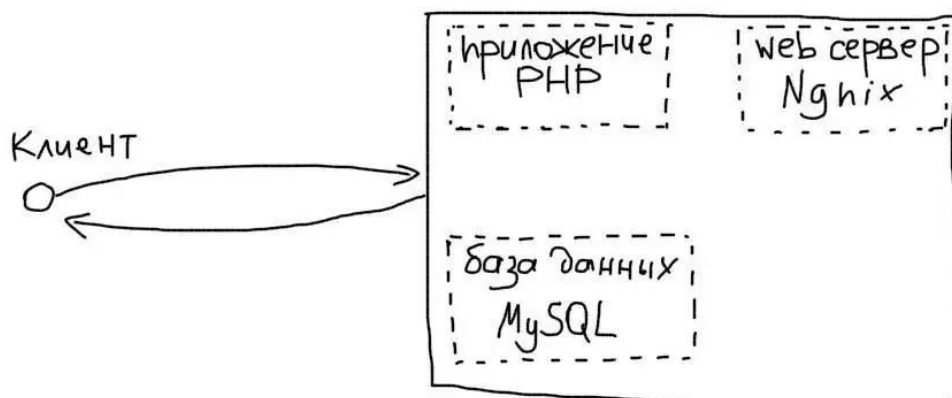


Рисунок 1.6 – Классическая архитектура PHP приложения.

Таким образом, выбирая в качестве языка программирования PHP, мы получаем огромные возможности по использованию языка, свободу выбора веб-сервера и операционной системы. PHP прекрасно подходит для разработки веб-приложений, но постоянным хранением информации не занимается. Всю информацию хранят сторонние сервисы – файловая система, различные базы данных и другие хранилища. PHP + СУБД - это мощная комбинация, позволяющая существенно упростить разработку веб-приложений и создавать более сложные веб-сайты [2].

Преимущества PHP:

- Легок в освоении на всех этапах
- Поддерживается большим сообществом пользователей и разработчиков
- Имеет развитую поддержку баз данных
- Имеется огромное количество библиотек и расширений языка
- Может использоваться в изолированной среде
- Предлагает средства организации веб-сессий, программный интерфейс расширений

- Портирован под большое количество аппаратных платформ и операционных систем
- Поддержка большинства возможностей ООП

Недостатки РНР:

- Не подходит для создания системных приложений или системных компонентов
- Глобальные параметры конфигурации влияют на базовый синтаксис языка, что затрудняет настройку сервера и ставит приложения в зависимость от конкретного окружения
- Большое количество устаревших и дублирующих функций, оставленных для совместимости со старыми версиями, смесь процедурных и объектных подходов, отсутствие единого стандарта наименования функций и параметров
- Невозможность многопоточной и асинхронной работы (отчасти решается сторонними модулями)
- Веб-приложения, написанные на РНР, зачастую имеют проблемы с безопасностью

Язык Ruby - интерпретируемый язык высокого уровня для быстрого и удобного объектно-ориентированного программирования. Язык обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, «сборщиком мусора» (механизм, обеспечивающий механическое управление памятью без вмешательства программиста) и многими другими возможностями [3].

Особенности языка:

- Простое и быстрое программирование
- Отсутствие необходимости объявления переменных
- Переменные не типизированы
- Простой и последовательный синтаксис
- Автоматическое управление памятью

- Объектно-ориентированное программирование
- Всё является объектом
- Классы, наследование, методы
- Mixin при помощи модулей (как и в PHP, множественное наследование в Ruby не поддерживается, но может быть достигнуто при помощи mixins – «примесей»)
- Итераторы и скобочные операции
- Целые числа с различной разрядностью
- Модель обработки исключительных ситуаций
- Динамическая загрузка
- Потоки
- Возможность прямого осуществления системных вызовов
- Мощная поддержка операций со строками и регулярными выражениями
- Немедленная обратная связь во время разработки

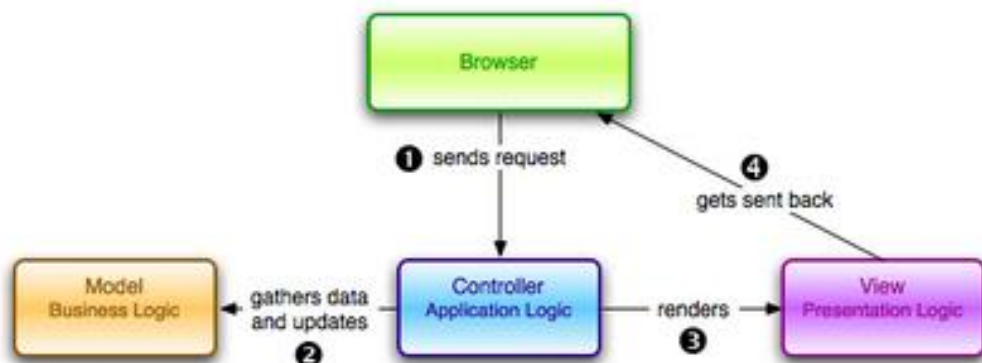


Рисунок 1.7 – Классическая архитектура Ruby приложения.

Преимущества Ruby:

- Реализует концептуально чистую объектно-ориентированную парадигму
- Предоставляет продвинутые методы манипуляции строками и текстом

- Легко интегрирует в свои программы высокопроизводительные серверы баз данных (DB2, MySQL, Oracle и Sybase)
- Простой и чистый синтаксис
- Имеет простой программный интерфейс для создания многопоточных приложений
- Имеет продвинутые средства для работы с массивами
- Возможности языка можно расширить при помощи библиотек, написанных на C или Ruby
- Зарезервированные слова могут являться идентификаторами, если это не создаёт неоднозначности для парсера
- Встроенный отладчик

Недостатки Ruby:

- Высокая сложность изучения языка
- Информационных ресурсов, посвящённых Ruby, недостаточно
- Ruby менее производителен по сравнению со многими другими языками, применяемыми в веб-разработке
- Ruby относительно медленно разрабатывается и развивается

Язык программирования Python используется для решения широкого спектра проблем и задач, в том числе для создания веб-сайтов.

Элегантный дизайн и эффективный, дисциплинирующий синтаксис этого языка облегчают программистам совместную работу над кодом. Python – язык, возможности которого изначально предполагалось унаследовать от нескольких, разных языков программирования, что позволяет совмещать процедурный подход к написанию кода с объектноориентированным и функциональным.

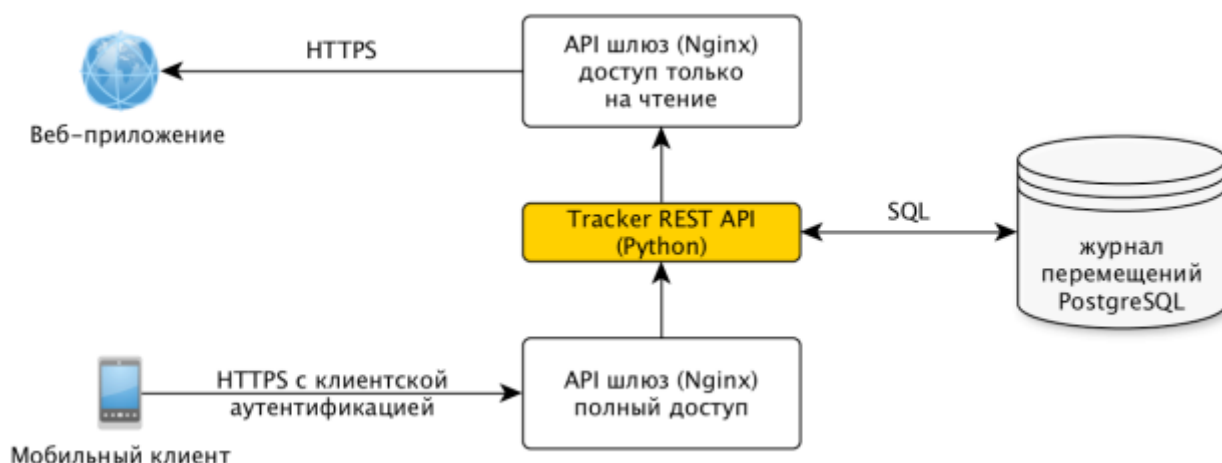


Рисунок 1.7 – Классическая архитектура Python приложения.

С точки зрения веб-разработки Python предоставляет простые и удобные методы для генерирования набора статических HTML-страниц, которые обслуживаются веб-сервером. Например, сценарий может принимать некоторое содержимое и добавлять к нему элементы оформления, типичные для веб-сайта (боковую панель, рекламу, стили и т.д.). Все это фактически сводится к работе с файлами и обработке текста. Скрипты написанные на языке программирования Python, могут применяться для создания динамических сайтов, т.е. в связке СУБД + JavaScript (Ajax) + Python. Сайт может работать на базе обычного вебсервера, такого как Apache, и отправлять данные из форм и запросов сценарию на языке Python для обработки и возвращения результата работы

Преимущества Python:

- Хорошо читаемый код вследствие особенностей синтаксиса
- Предоставляет средства быстрого прототипирования (создание опытных образцов или работающих моделей системы) и динамической семантики (конструкция языков программирования)
- Большое русское сообщество

- Множество полезных библиотек и расширений языка, которые можно легко использовать в проектах благодаря предельно унифицированному механизму импорта и программным интерфейсам
 - Хорошо продуманные и легкие в использовании механизмы модульности (функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом)
 - Абсолютно всё в Python является объектами в смысле ООП, но при этом объектный подход не навязывается программисту
 - При высоких нагрузках на приложения использующих Python, производительность будет выше, чем в приложениях на Ruby, Java или PHP
- Недостатки Python:

Несовершенная поддержка многопоточности

- На Python создано не так уж много качественных программных проектов по сравнению с другими универсальными языками программирования, например, PHP, Java или Ruby
- Изначальная ограниченность средств для работы с базами данных

Даже после сравнения языков выбор одного из них, несомненно, является сложной задачей, поскольку они хороши каждый на своем месте. Python используется для машинного обучения. Ruby отлично реализует серверные приложения.

Выбор остановлен на PHP как на универсальном инструменте. На PHP реализуются серверные скрипты и элементы веб-разработки.

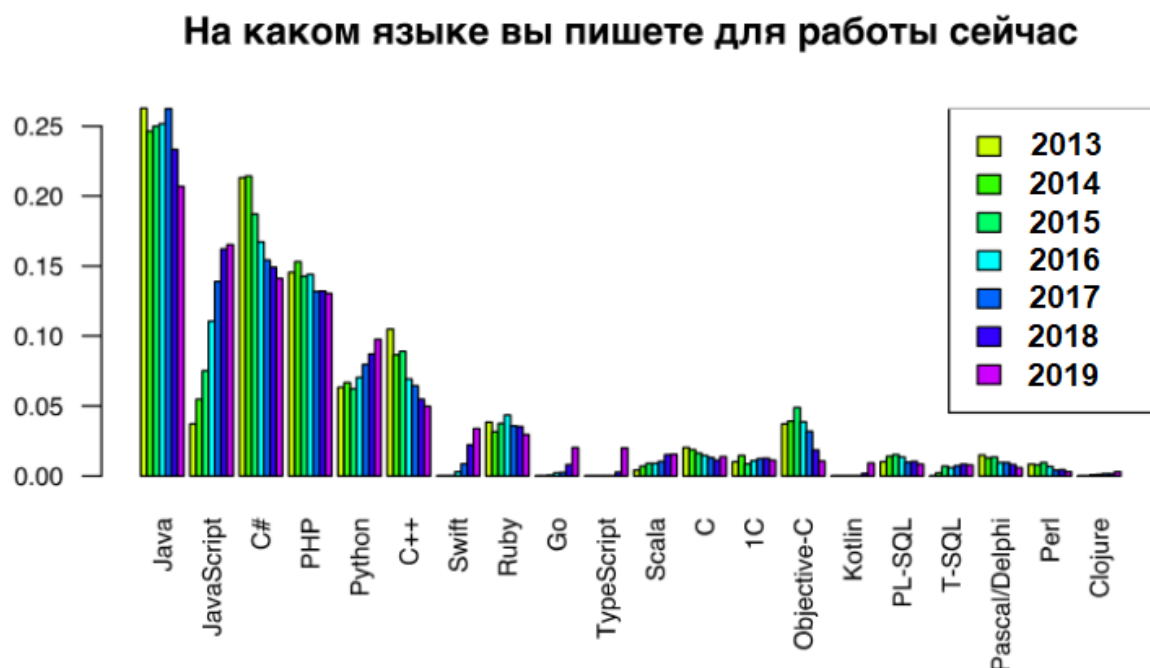


Рисунок 1.7 – Популярные инструменты реализации системы e-commerce.

1.6.2 Выбор вспомогательных элементов разработки

В данном разделе опишем инструменты разработки, которые будут использоваться для реализации проект.

jQuery - набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

Возможности jQuery:

- Движок кросс-браузерных CSS-селекторов Sizzle, выделившийся в отдельный проект;
- Переход по дереву DOM, включая поддержку XPath как плагина;
- События;
- Визуальные эффекты;
- AJAX-дополнения;
- JavaScript-плагины.

Handlebars (шаблонизатор в web) - программное обеспечение, позволяющее использовать html-шаблоны для генерации конечных html-страниц. Основная цель использования шаблонизаторов - это отделение представления данных от исполняемого кода. Часто это необходимо для обеспечения возможности параллельной работы программиста и дизайнера-верстальщика. Использование шаблонизаторов часто улучшает читаемость кода и внесение изменений во внешний вид, когда проект целиком выполняет один человек.

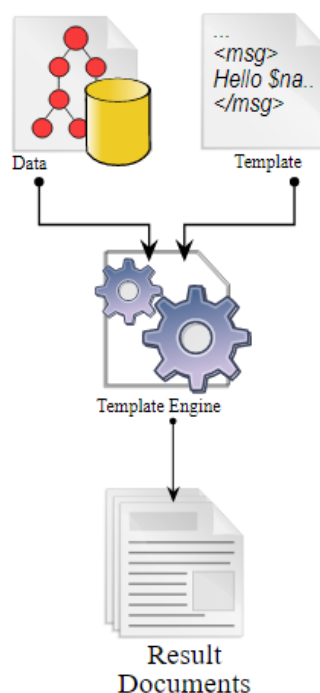


Рисунок 1.8 – Шаблонизатор JavaScript

1.7 Выбор платежной системы для реализации оплаты в разрабатываемой системе

Платежная система PayPal имеет на сегодняшний момент большинство инструментов для реализации платежного инструмента интернет магазина: гибкий инструмент выставления счетов, инструменты электронной коммерции, контроль процесса оплаты, инструменты отчетности. Ниже приведено сравнение данной платежной системы с конкурентами.

Для объективности сравнения возьмем выборку наиболее популярных платежных систем 2019 года: Payline, Shopify, TransferWise, 2Checkout, Payoneer, Skrill, Google Pay, Authorize.Net, Dwolla. Braintree, WorldPay, Amazon Pay, Klarna, WePay.

1.7.1 Система Payline

Payline это платежный процессор, который предлагает широкий спектр решений, в том числе мобильные, онлайн и платежи в магазине.

При реализации банковских транзакций транзакций, Payline пока уступает PayPal.

Плюсы:

- Payline предлагает широкий спектр пакетов обработки кредитных карт в магазине, в комплекте с надежным вспомогательным оборудованием.
- Плата за транзакции оффлайн Payline дешевле, чем PayPal.
- Он предлагает полнофункциональный API для облегчения интеграции со сторонними решениями и платформами.
- Payline поддерживает мобильные платежи.

Минусы:

- Пока система Payline доступна только в США.
- Трудно предсказать, какие сборы понесет клиент с помощью модели обмена и ценообразования.
- Функции обработки платежей в электронной торговле Payline не могут совпадать с функциями PayPal.

1.7.2 Система Shopify

Shopify - встроенная система обработки платежей, которая поставляется по умолчанию как системы e-commerce Shopify.

Разработчикам не нужно беспокоиться о встраивании внешнего платежного шлюза. Разработчик настраивает управление системой обработки

транзакций прямо из Панели приборов. Конечным результатом настройки является связь между интернет-магазином и платежной системой.

Плюсы

- Встроенный в Shopify инструмент управления платежами.
- Shopify может быть интегрирован с широким спектром платформ и решений, включая сторонние платежные системы.
- Бесперебойная работа с многочисленными бухгалтерскими приложениями.
- Минимальные комиссии за транзакции на Shopify.

Минусы

- Shopify доступны только в странах европы.
- Аккаунт может быть заблокирован без предупреждения.
- Shopify вычитают \$15 за каждый возвратный платеж.

1.7.3 Система TransferWise

Система TransferWise разрабатывалась как альтернатива PayPal для международных переводов.

Плюсы:

- Один из самых дешевых инструментов для международных денежных переводов, с клиентами по всему миру.
- Прозрачное ценообразование.
- Быстрая конвертация валют.
- Возможность выставять счета в валюте клиента.

Минусы:

- Работа только с банковскими переводами.
- Работа только со счетами в EUR, USD, AUD и GBP.
- Ограничения по количеству аккаунтов и функций

1.7.4 Система Square

Платежная система Square - это не просто процессор регулярных платежей. Он предлагает интернет-магазин плюс домен бесплатно, а затем объединяет его с интеграцией корзины покупок, а также выставлением счетов, хранилищем информации о карте и функциями виртуального терминала. А для обработки карт вы получаете POS и mPOS бесплатно.

Плюсы

- Поставляется с бесплатным доменом и интернет-магазином.
- Square - это комплексное платежное решение «все в одном».
- Существуют дополнительные услуги для бизнеса.
- Square не взимает никаких ежемесячных платежей.
- Система выгодна для торговцев с низким объемом.

Минусы

- В качестве платежного шлюза Square доступен только в США, Великобритании, Канаде, Австралии и Японии.
- Является сторонним платежным агрегатором, учетные записи которого могут быть использованы третьими лицами.

Система PayPal будет выбрана в данном проекте в связи с тем, что лишена проблемы совместимости с современными языками программирования и СУБД.

В системе PayPal проработаны опции и API для личной отправки и для платежей в электронной торговле. В тоже время она адаптирована для банковских переводов и позволяет совершать транзакции в любой стране. То есть на сегодняшний момент система PayPal обладает универсальным инструментарием для реализации функционала проекта.

1.8 Выводы по главе

В первой главе дипломного проекта:

были определены функциональные требования к реализуемой системе
определены задачи, которые будет решать система для пользователей

					ПГУ 2.090301.36.001 ПЗ	Лист
						26
Изм.	Лист	№ докум	Подпись	Дата		

обоснованы и выбраны основные и вспомогательные средства разработки системы

определены инструменты СУБД и платежной системы для работы в рамках приложения.

					ПГУ 2.090301.36.001 ПЗ	Лист
						27
Изм.	Лист	№ докум	Подпись	Дата		

2. МЕТОДЫ И СРЕДСТВА РЕШЕНИЯ ЗАДАЧИ

2.1 Требования к функциональному назначению приложения

Программа должна поддерживать работу разных типов пользователей.

Возможности, которые должны быть реализованы в зависимости от пользователя, представлены в таблице 3.

Таблица 3 – Возможности пользователей

Тип пользователя	Возможности
Администратор	Формирует списки каталогов и товаров. Связывает товар с определенным каталогом. Привязывает к товару купоны скидок. Просматривает заказы и статистику заказов.
Покупатель	Производит просмотр содержимого каталогов. Осуществляет поиск товара по критерию. Осуществляет добавление товара в корзину. Осуществляет платеж.

Доступ к возможностям системы имеют как авторизированные пользователи, так и гости.

В соответствии с основными функциями пользователей были разработаны диаграммы для каждого типа пользователей сайта. Диаграммы представлены в приложении.

Первым делом в реализации системы проведем проектирование системы и затем произведем кодирование и тестирование реализованного программного продукта.

2.2 Проектирование системы

Проектирование играет важную роль в любом проекте разработки программного обеспечения. Это гарантирует полноту и правильность программной системы и оправдывает ожидания конечных пользователей. Кроме того, Проектирование служит единственной контрольной точкой для перекрестной проверки требований перед кодированием.

На рисунке 2.1 показан аппаратно программный комплекс реализации проекта.

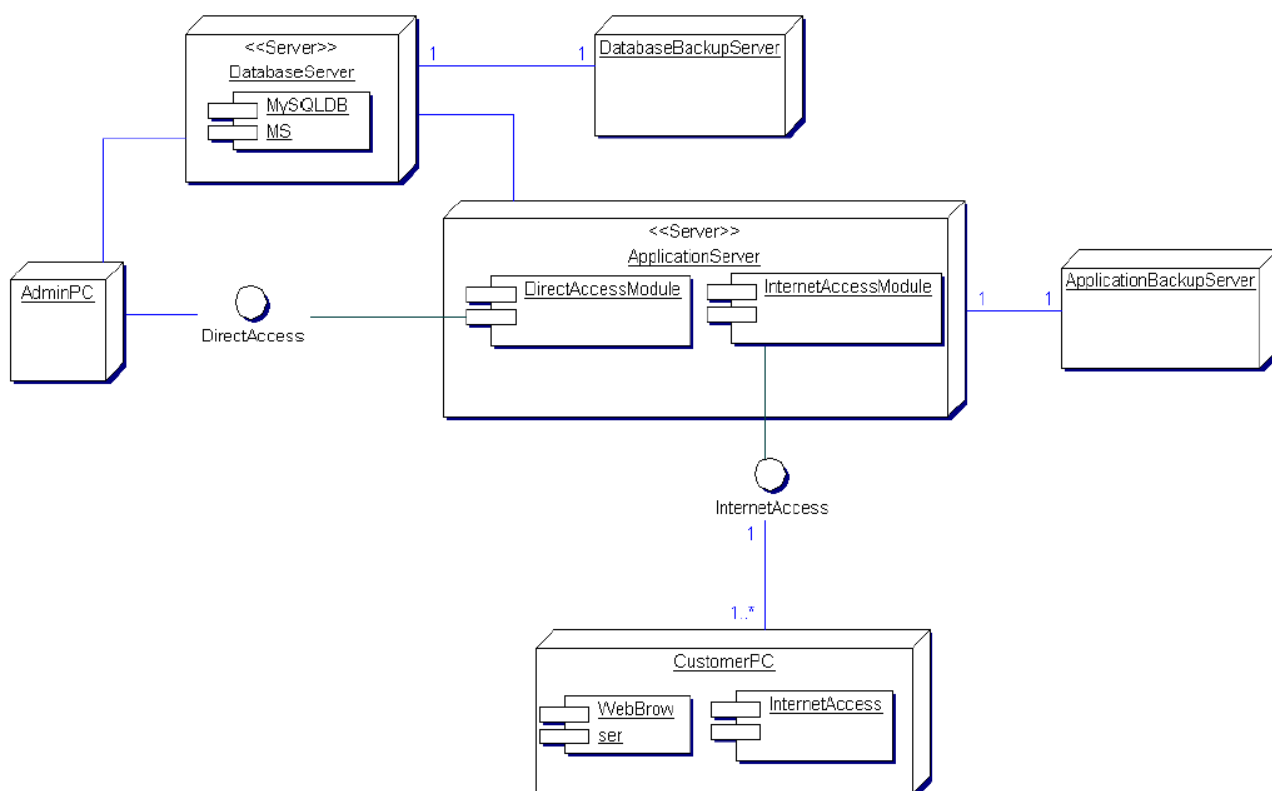


Рисунок 2.1 Система аппаратно-программной платформы

Как видно из UML диаграммы система состоит из четырех связанных между собой частей: АРМ администратора системы, АРМ пользователя системы, сервера приложения, сервера СУБД (включает в себя базу данных и систему резервного копирования). Данная архитектура полностью

соответствует ранее выбранному паттерну MVC, на основании которого реализуется приложение.

С помощью диаграмм использования проведем описание работы администратора и пользователя системы. Данная работа позволит реализовать диаграмму классов системы, правильно разбить ее модели, вид и контроллеры.

Исходя из требований, описанных в первой главе администратору необходимо обязательно иметь возможность авторизованного входа в систему, а также наличие функций (рисунок 2.2).

Как показано на рисунке администратор после авторизации может: добавить категорию товара и/или конкретный продукт, редактировать категорию товара и/или конкретный продукт, удалить категорию товара и/или конкретный продукт, просмотр списка категорий товара и/или конкретного продукта с сохранением данных в СУБД, просмотр оплаты, просмотр подписчиков, просмотр заказов, детализация информации о заказе, обновление информации о заказе.

Пользователю не обязательно иметь возможность авторизованного входа в систему для реализации покупки, но такая возможность ему будет дана для расширения его возможностей в рамках системы. Пользователь обладает функциями, показанными на рисунке 2.3.

Пользователь реализует: просмотр наличия товаров на площадке, добавление товара в корзину, обновление корзины, удаление продукта из корзины, единоразовая операция удаления всех товаров из корзины, возможность оплаты товара через платежную систему, возможность просмотра заказа перед оплатой, возможность оформления подписки на рассылки.

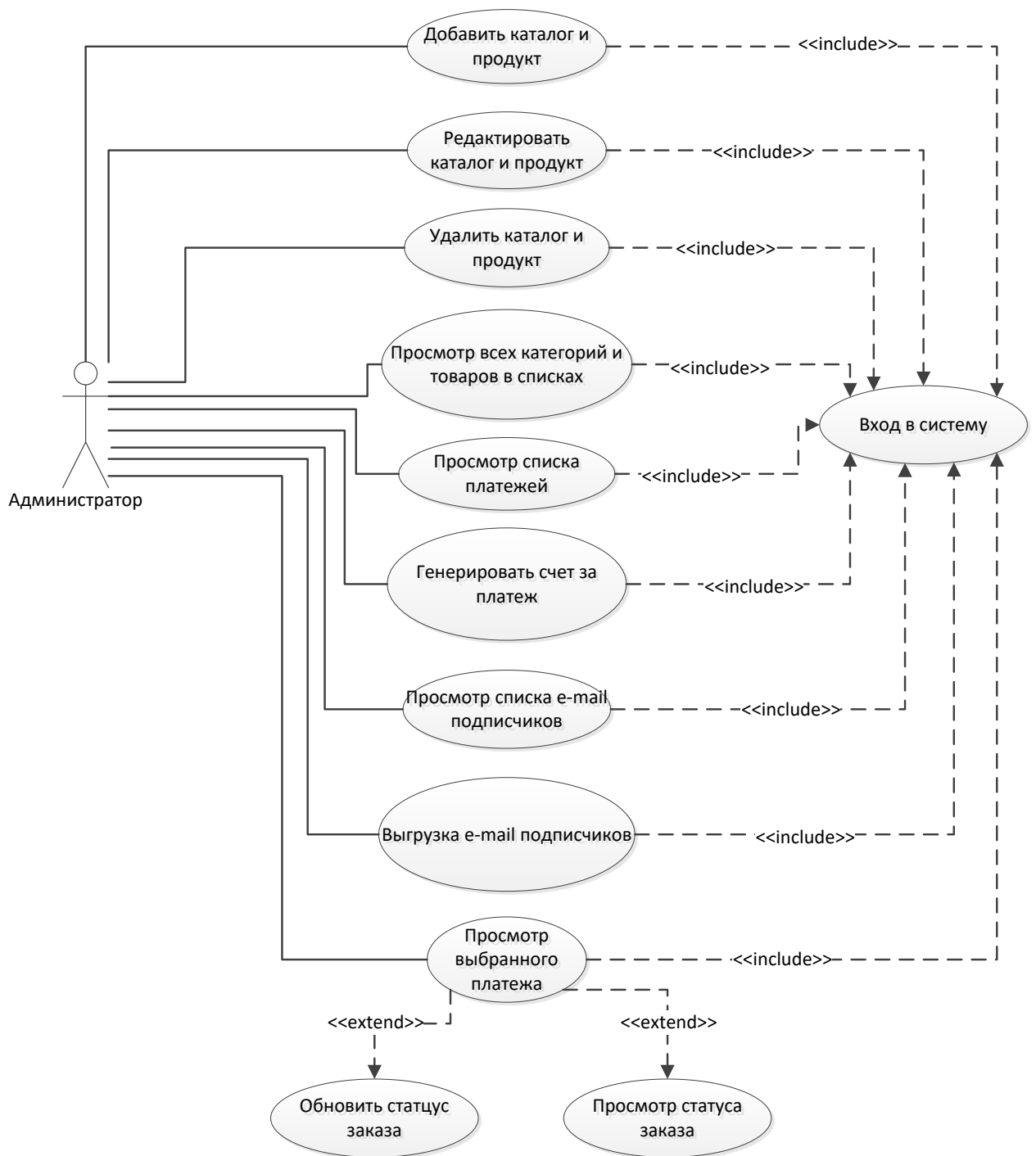


Рисунок 2.2 Описание функций администратора в формате UML

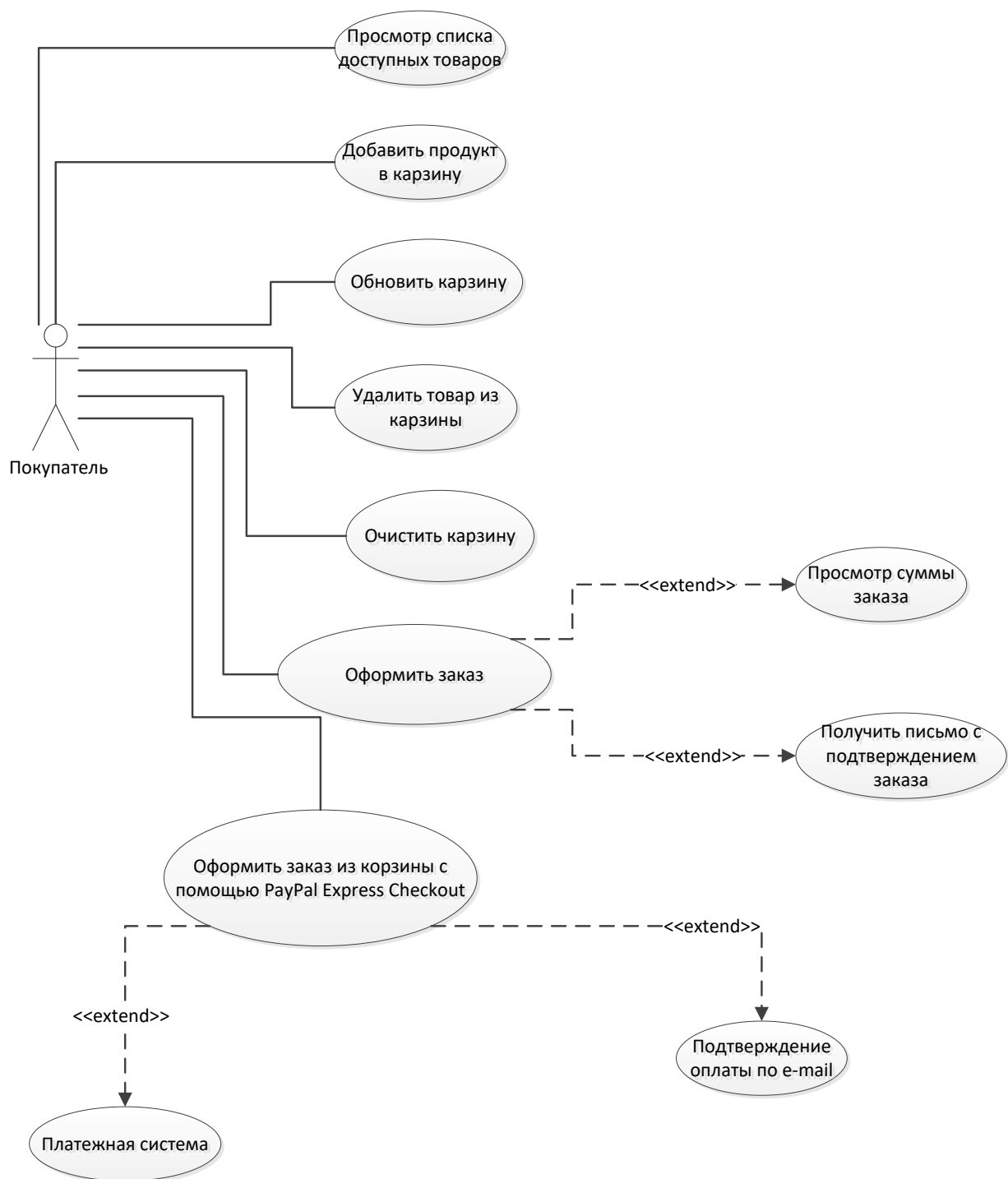


Рисунок 2.3 Описание функций покупателя в формате UML

На основании функций пользователей системы описанных в рамках Use Case диаграмм проведем описание компонентов каждой части MVC системы в категориях компонентов.

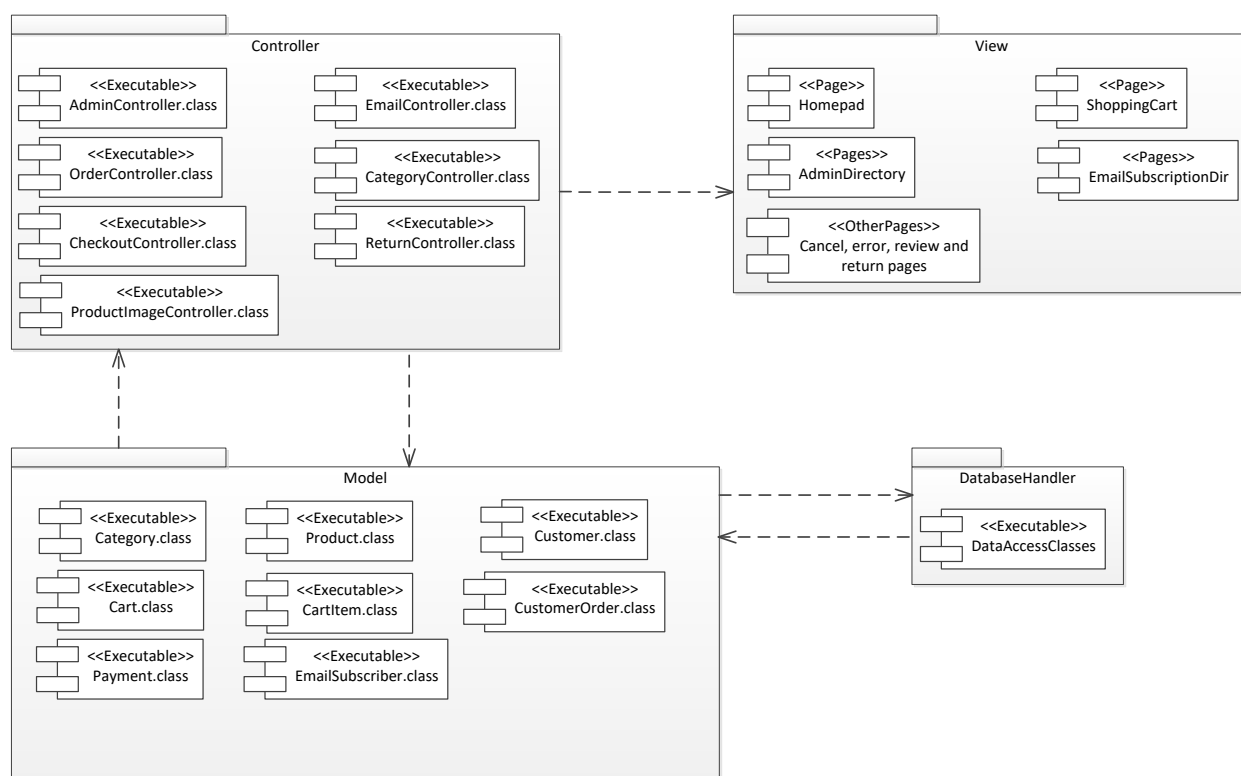


Рисунок 2.4 Диаграмма развертывания MVC системы

Как видно из рисунка 2.4 блок системы: Слой вид приложения представлен компонентами домашней страницы, панели администрирования, карточки категории и/или товара, страницы оформления подписки и страницы обработки ошибок и/или исключительных ситуаций.

Слой контроллер состоит из модулей обработки запросов администратора (AdminController), модуля обработки заказа (OrderController), модуля работы с категориями товара (CategoryController), модуля работы с возвратом товара из корзины (ReturnController), модуля оплаты товара (CheckoutController), модуля соответствия описания товара и фотографий для него (ProductImageController).

Слой модель системы связана с обработчиком СУБД (DatabaseHandler) и состоит из модулей для хранения и передачи следующей категории информации: категории товаров (Category), наименования конкретных товаров (Product), информации о покупателях (Customer), информации о корзине (Cart), и товарах в ней (CartItem), информации о заказе

(CustomerOrder), платежах за товар (Payment), информации о подписчиках (EmailSubscriber).

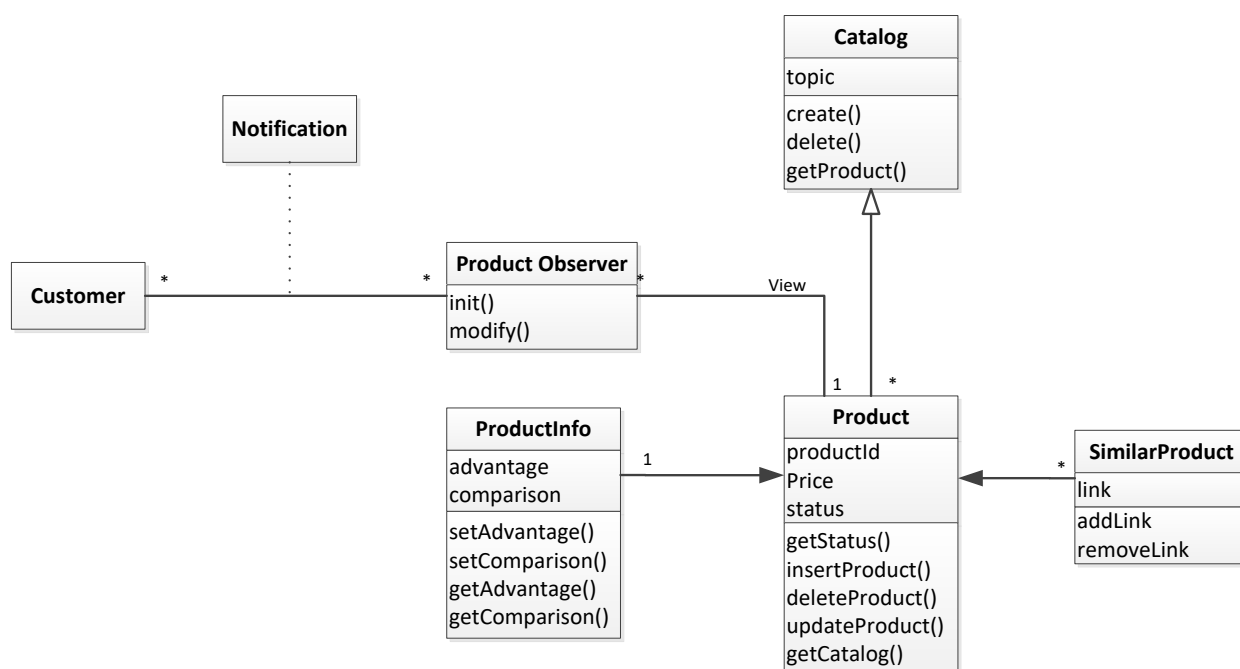


Рисунок 2.5 Диаграмма классов реализации работы с каталогом товаров (формирования каталога).

Класс Catalog является точкой сбора продуктов для формирования корзины. Различные категории товаров и описания связанных продуктов определены в отдельных классах. Для поиска изменений и уведомления клиентов об изменениях при операциях с товарами используется шаблон Observer.

Класс Catalog - это коллекция товаров. Каждый товар принадлежит как минимум одному каталогу. Класс Product определяет тип продаваемого товара, который покупатели могут добавить в корзину. Класс Product содержит основные атрибуты каждого продукта. В частности, атрибут статуса указывает на особые маркеры, как например то, что продукт является новым (определяется датой добавления).

Класс ProductInfo предоставляет подробную информацию о продукте. Сюда также может входить сравнение разных типов одного и того же

продукта, разных брендов или предоставление наилучшего соотношения цена/качество. Например, для товара объект класса ProductInfo может предоставлять сравнительную информацию от нескольких брендов. Класс SimilarProduct предоставляет ссылки на другие похожие продукты.

Изменения в продуктах уведомляются клиентам по электронной почте, чтобы сообщить им, что есть какой-то новый или интересный продукт. Класс ProductObserver следит за изменениями и уведомляет клиентов. Все эти классы являются типом шаблона Observer. Класс Notification ведет учет уведомлений, отправляемых пользователям.

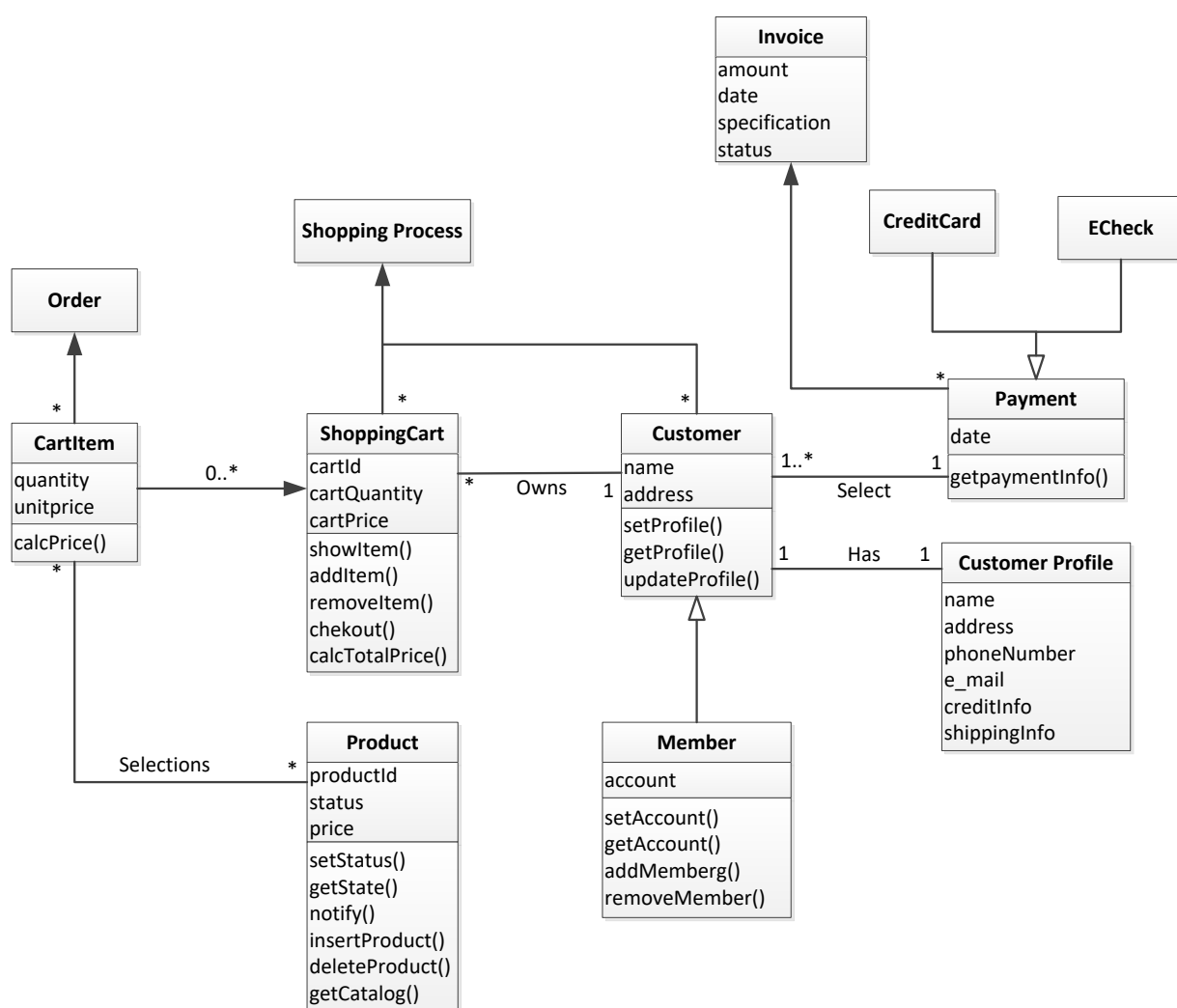


Рисунок 2.6 Диаграмма классов процесса покупки товара.

Часть классов на рисунке 2.6 формирует структуру, необходимую для выбора и покупки товара в системе.

Т.к. СУБД содержит большой перечень товаров, клиенты могут выбирать и покупать несколько товаров в одном сеансе. Клиент ищет интересующий его продукт, выбирает что-то и добавляет в свой список покупок. Он может проверять и модифицировать корзину. Клиенты могут быть зарегистрированы на площадке или использовать ее в качестве гостя.

В результате выше сказанного были реализованы классы бизнес логики которые позволяют процесс покупки основать на концепции корзины покупок, аналогичной тележкам, используемым в супермаркетах. У покупателя может быть несколько заказов, и одна общая корзина содержащая его выбор. Заказ создается, когда покупатель решает купить (оформить заказ) и отправляет его в свою корзину. Счет выставляется за весь оформленный заказ в корзине.

Класс `ShoppingProcess` является единой точкой входа для всего процесса покупки. Класс `ShoppingCart` собирает информацию обо всех продуктах, которые выбрал покупатель. Объект `CartItem` указывает количество и продукт, выбранный клиентом.

Клиент может запросить товары в своей корзине и удалить товары из корзины. Класс `Customer` указывает покупателя, ответственного за корзину. Этот класс также предоставляет операции, позволяющие клиенту изменять данные покупок (количество и номенклатуру товара). Система предоставляет несколько форм оплаты, например, кредитную карту и электронный чек. Когда корзина будет оформлена, будут сгенерированы заказ и счет.

Полный набор диаграмм работы системы с точки зрения администратора (рисунок 2.7) и пользователя (рисунок 2.8) представлен в приложении.

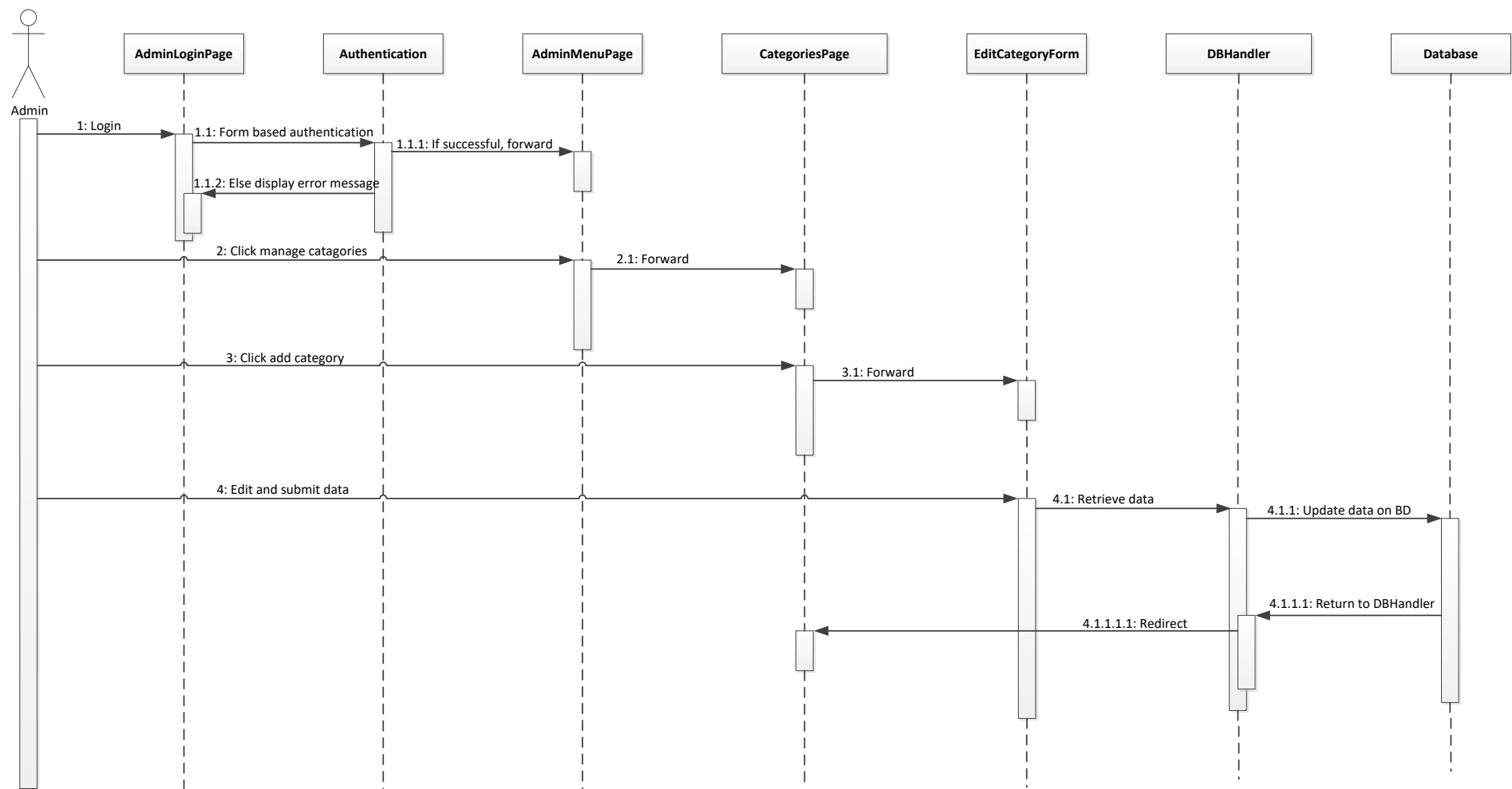


Рисунок 2.7 Диаграмма работы администратора в системе.

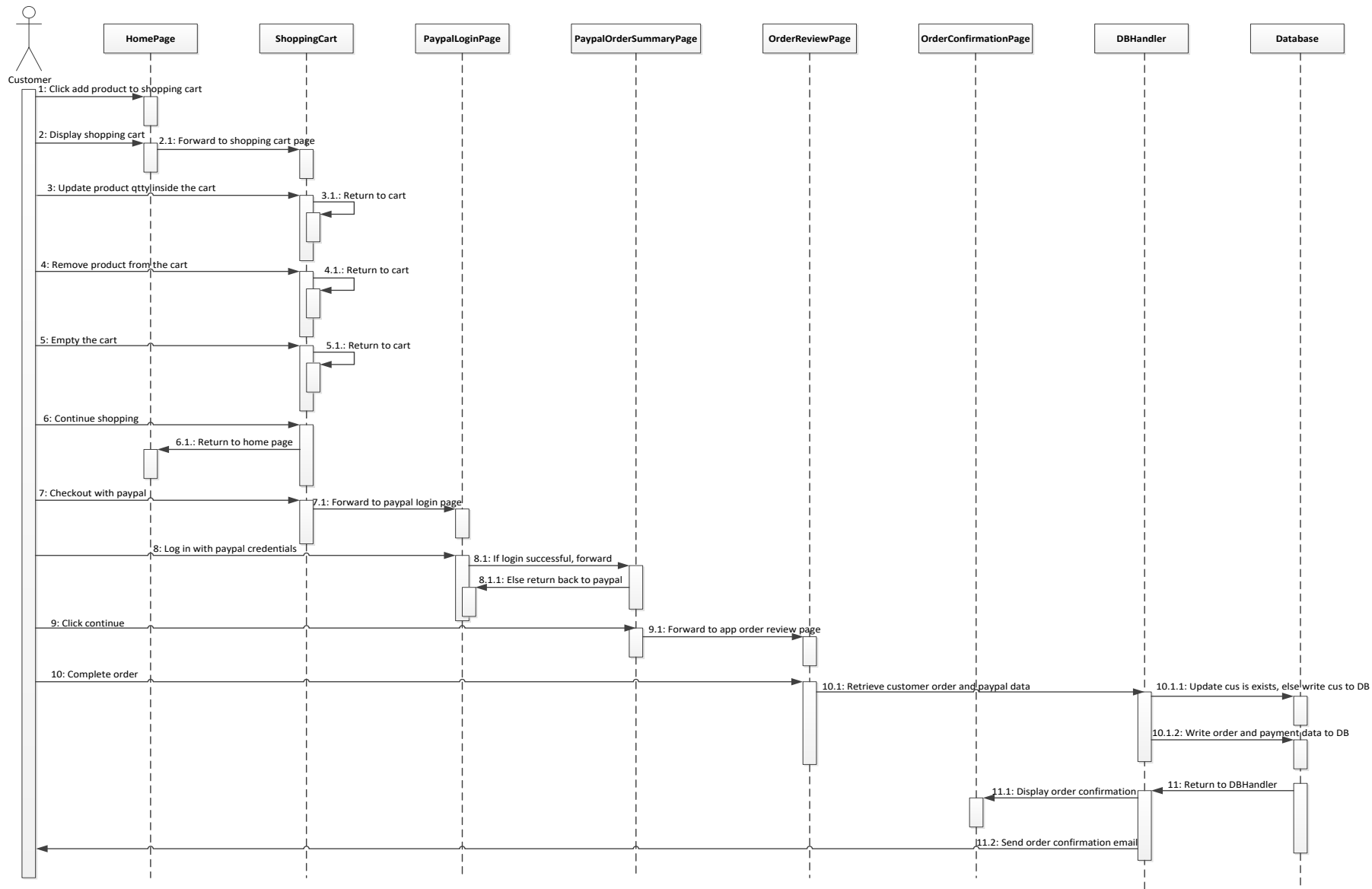


Рисунок 2.8 Диаграмма работы пользователя в системе.

2.3.Проектирование структур данных

2.3.1 Концептуальное проектирование

Процесс построения инфологической модели состоит из следующих шагов:

- определение сущностей;
- определение зависимостей между сущностями;
- задание первичных и альтернативных ключей;
- определение атрибутов сущностей.

Далее в таблице 4 приведен обзор сущностей, где представлены все спроектированные сущности БД, заголовки столбцов означают следующее:

- название сущности – название выделенной в предметной области сущности;
- идентификация таблицы – физическое имя сущности;
- количество записей – предлагаемое количество записей сущности;
- описание – описание сущности.

Таблица 4– Таблица сущностей

Название	Идентификатор	Описание
Пользователи	users	Информация о пользователях
Товары	product	Информация о товарах
ТоварыДоп	product_meta	Дополнительная информация о товаре
ТоварыОтзывы	product_review	Отзывы о товаре
Теги	tag	Информация о тегах товаров
Корзина покупок	cart	Информация о корзине пользователя
ТоварыКорзины	Cart_item	Информация о товарах в корзине

Продолжение Таблицы 4

Название	Идентификатор	Описание
Заказ	order	Информация о заказах
ТоварыЗаказа	orderitem	Информация о заказанных товарах
Платежи	transact	Информация о платежах
Категория	category	Каталог товара

Логическое проектирование включает в себя следующие этапы:

- определение набора отношений исходя из концептуальной модели БД;
- проверка модели с помощью правил нормализации;
- проверка модели в отношении модели пользователей;
- создание полной атрибутивной диаграммы в рамках технологии IDEF1X;
- определение требующейся поддержки целостности.

Анализ функциональных зависимостей в процессе логического проектирования для каждого отношения приведен ниже.

Отношение: Пользователи.

Схема отношения: Заказы, Товары, Платежи, Корзина

Первичный ключ: идентификатор пользователя.

Альтернативный ключ: мобильный номер телефона пользователя.

Функциональные зависимости:

Идентификатор пользователя → идентификатор платежа,
идентификатор корзины.

Отношение находится в 3НФ.

Отношение: Товары.

Схема отношения: Категории, Товары Отзывы, ТоварыДоп , Товары
Корзины

Первичный ключ: идентификатор товара.

Альтернативный ключ: нет.

Функциональные зависимости:

Идентификатор товара → идентификатор пользователя, идентификатор
корзины.

Отношение находится в ЗНФ.

Отношение: ТоварыДоп.

Схема отношения: Товары

Первичный ключ: идентификатор характеристики товара.

Альтернативный ключ: нет.

Функциональные зависимости:

Идентификатор характеристики товара → идентификатор товара.

Отношение находится в ЗНФ.

Отношение: ТоварыОтзывы.

Схема отношения: Товары

Первичный ключ: идентификатор характеристики товара.

Альтернативный ключ: нет.

Функциональные зависимости:

Идентификатор характеристики товара → идентификатор товара.

Отношение находится в ЗНФ.

Отношение: Тэги

Схема отношения: Товары

Первичный ключ: идентификатор тега.

Альтернативный ключ: тег (он уникален для системы).

					ПГУ 2.090301.36.001 ПЗ	Лист
						41
Изм.	Лист	№ докум	Подпись	Дата		

Функциональные зависимости:

Идентификатор тега → идентификатор товара.

Отношение находится в 3НФ.

Отношение: КорзинаПокупок

Схема отношения: Товары Корзины, Пользователь

Первичный ключ: идентификатор товара в корзине.

Альтернативный ключ: пользователь привязанный к корзине

Функциональные зависимости:

Идентификатор корзины покупок → идентификатор товара,
идентификатор пользователя.

Отношение находится в 3НФ.

Отношение: Заказ

Схема отношения: Платежи, Пользователь, ТоварыЗаказа

Первичный ключ: идентификатор заказа.

Альтернативный ключ: пользователь привязанный к заказу

Функциональные зависимости:

Идентификатор заказа → идентификатор товара, идентификатор
пользователя.

Отношение находится в 3НФ.

Отношение: Платежи

Схема отношения: Заказ, Пользователь

Первичный ключ: идентификатор платежа.

Альтернативный ключ: пользователь привязанный к платежу

Функциональные зависимости:

Идентификатор платежа → идентификатор заказа, идентификатор
пользователя.

Отношение находится в 3НФ.

2.3.2 Даталогическое проектирование

На данном этапе разработки БД была сформирована физическая модель БД, изображенная в таблицах 5 -13.

Таблица 5 – Отношение «Пользователь» («User»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Id	Идентификатор пользователя	int	11	-	PrimaryKey
First Name	Имя	Varchar	50	-	Attribute
Middle Name	Отчество	Varchar	50	-	Attribute
Last Name	Фамилия	Varchar	50	-	Attribute
Mobile	Телефон	Varchar	25		Attribute
Email	Почта	Varchar	50		Attribute
Password Hash	Пароль	Varchar	32		Attribute
Admin	Администратор	TINYTEXT	1		Attribute
Vendor	Производитель	TINYTEXT	1		Attribute

Продолжение Таблицы 5

Имя столбца	Содержа-тельное описание	Тип данных	Размер ность	Возмож ность значени я Null	Роль
Registered At	Когда зарегистрировался	DATETIME			Attribute
Last Login	Последний вход	DATETIME			Attribute
Intro	Короткая информация о пользовтеле	TINYTEXT			Attribute
Profile	Расширенная информация	TEXT			Attribute

Таблица 6 – Отношение «Товары» («Product»)

Имя столбца	Содержа-тельное описание	Тип данных	Размер ность	Возмож ность значени я Null	Роль
Id	Идентификатор продукта	int	11	-	PrimaryKey
User Id	Имя	int	11	-	ForeingKey
Title	Наименование	Varchar	75	-	Attribute
Meta Title	Ter	Varchar	100		Attribute

Продолжение Таблицы 6

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Slug	Токен	Varchar	100		Attribute
Summary	Ключ тегов	TINYTEXT			Attribute
Type	Тип	SMALLINT	6		Attribute
SKU	Склад	Varchar	100		Attribute
Price	Цена	FLOAT			Attribute
Discount	Скидка	FLOAT			Attribute
Quantity	количество	SMALLINT	6		Attribute
Shop	Доступность товара	TINYTEXT			Attribute
Created At	Создан	DATETIME			Attribute
Updated At	Обновлен	DATETIME			Attribute
Published At	Опубликован	DATETIME			Attribute
Starts At	Дата продаж	DATETIME			Attribute
Content	Информация о продукте	TEXT			Attribute

Таблица 7 – Отношение «ТоварыДон» («*ProducMetat*»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Id	Идентификатор доп. записи продукта	int	11	-	PrimaryKey
Product Id	Идентификатор продукта	int	11		ForeingKey
Key	Краткая доп. информация	Varchar	50		Attribute
Content	Расширенная доп. информация	TEXT			Attribute

Таблица 8 – Отношение «ТоварыОтзывы» («*ProducReview*»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Id	Идентификатор отзыва	int	11	-	PrimaryKey
Product Id	Идентификатор продукта	int	11		ForeingKey

Продолжение Таблицы 8

Имя столбца	Содержа-тельное описание	Тип данных	Размер ность	Возмо жност ь значен ия Null	Роль
Parent Id	Идентификатор отзыва	int	11		ForeingKey
Title	Заголовок	Varchar	100		
Rating	Рейтинг	int	11		
Published	Признак доступности	DATETI ME			
Created At	Создан	DATETI ME			
Published At	Опубликован	DATETI ME			
Content	Содержимое	TEXT			

Таблица 9 – Отношение «Категория» («Category»)

Имя столбца	Содержа- тельное описание	Тип данных	Размер ность	Возмо жност ь значен ия Null	Роль
Id	Идентификатор категории	int	11	-	PrimaryKey

Продолжение Таблицы 9

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Parent Id	Родительская категория	int	11		ForeingKey
Title	Заголовок	Varchar	100		
Meta Title	Мета теги для SEO	Varchar	100		
Content	Содержание	TEXT			

Таблица 10 – Отношение «*ПродуктКатегория*» («*ProductCategory*»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
ProductId	Идентификатор категории	int	11	-	PrimaryKey
CategoryId	Родительская категория	int	11		ForeingKey

Таблица 11 – Отношение «Корзина» («Cart»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Id	Идентификатор категории	int	11	-	PrimaryKey
UserId	пользователь	int	11		ForeingKey
SessionId	Сессия	int	11		ForeingKey
Token	Платежный шлюз	int	11		
Status	Статус	SMALLINT	6		
First Name	Имя	Varchar	100		
Middle Name	Отчество	Varchar	100		
Last Name	Фамилия	Varchar	100		
Mobile	Телефон	Varchar	100		
Email	Почта	Varchar	100		
Line 1	Адрес 1	Varchar	100		
Line 2	Адрес 2	Varchar	100		
City	Город	Varchar			
Province	Область	Varchar			
Country	Страна	Varchar			

Продолжение Таблицы 11

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Created At	Создан	DATETIME			
Updated At	Обновлен	DATETIME			
Content	Доп. информация	TEXT			

Таблица 12 – Отношение «Заказ» («Cart»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Id	Идентификатор заказа	int	11	-	PrimaryKey
User Id	Идентификатор пользователя	int	11	-	ForeingKey
Session Id	Идентификатор сессии	int	11	-	ForeingKey
Token	Токен платежного шлюза	int	11		
Status	Статус	SMALLINT	6		
Sub Total	Цена заказа	FLOAT			

Продолжение Таблицы 12

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Item Discount	Скидка на единицу	FLOAT			
Tax	НДС	FLOAT			
Shipping	Доставка	FLOAT			
Total	Сумма	FLOAT			
Promo	Промо код	FLOAT			
Discount	Скидка на все	FLOAT			
Grand Total	Окончательная сумма	FLOAT			
First Name	Имя	Varchar	100		
Middle Name	Отчество	Varchar	100		
Last Name	Фамилия	Varchar	100		
Mobile	Телефон	Varchar	50		
Email	Почта	Varchar	20		
Line 1	Адрес 1	Varchar	100		
Line 2	Адрес 2	Varchar	100		
City	Город	Varchar	100		
Province	Область	Varchar	100		

Продолжение Таблицы 12

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Country	Страна	Varchar	100		
Created At	Создан	DATETIME			
Updated At	Обновлен	DATETIME			
Content	Доп. информация	TEXT			

Таблица 13 – Отношение «Платежи» («Transact»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Id	Идентификатор заказа	int	11	-	PrimaryKey
User Id	Идентификатор пользователя	int	11		ForeingKey
Order Id	Идентификатор сессии	int	11		ForeingKey
Code	Id платежа для платежной системы	Varchar	100		
Type	Тип платежной карты	SMALLINT	6		

Продолжение Таблицы 13

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
Mode	Офлайн/онлайн	SMALLINT	6		
Status	Статус заказа	SMALLINT	6		
Created At	Создан	DATETIME			
Updated At	Обновлен	DATETIME			
Content	Детали	TEXT			

2.3.3 Инфологическая модель

В базах данных содержится информация об объектах, существующих в реальном мире. Все эти объекты являются материальными, обладающим некоторыми свойствами и присущим им поведением. При создании программного продукта, свойства объектов хранятся в базе данных, а их поведение реализуется в приложении посредством методов — процедур, которые представляют собой отклик на события инициируемых пользователем.

Инфологическая модель является проблемно-ориентированной и системно-независимой, т.е. не зависимой от конкретной СУБД, операционной системы и аппаратного обеспечения ЭВМ.

Основным требованием к инфологической модели, вытекающим из ее назначения, является требование адекватного отображения предметной области. Инфологическая модель должна быть непротиворечивой. Она является единым интегрированным описанием предметной области и отражает взгляды и потребности всех пользователей системы.

Инфологическая модель представляет информационные потоки, сущности и связи данной предметной области. Она может быть представлена в виде ER-модели и реляционной схемы.

Целью построения этой структуры является выявление и объединение информационных требований пользователя, связей между элементами данных без относительно к их содержанию и среде их хранения. Инфологическая модель должна обладать свойством легкой расширяемости, обеспечивающим ввод новых данных без изменений ранее определенных.

Центральной компонентой инфологической модели является описание объектов предметной области и связей между ними (ER-модель).

Цель инфологического моделирования — обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком (последний не может быть использован в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка).

Основными конструктивными элементами инфологических моделей являются сущности, связи между ними, идентификаторы (ключи) и свойства (атрибуты).

Сущность — любой различимый объект (объект, который мы можем отличить от другого), информацию о котором необходимо хранить в базе данных. Сущностями могут быть люди, места, самолеты, рейсы, вкус, цвет и т.д. Необходимо различать такие понятия, как тип сущности и экземпляр сущности. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое.

Инфологическая модель предметной области приведена на рисунке 2.9.

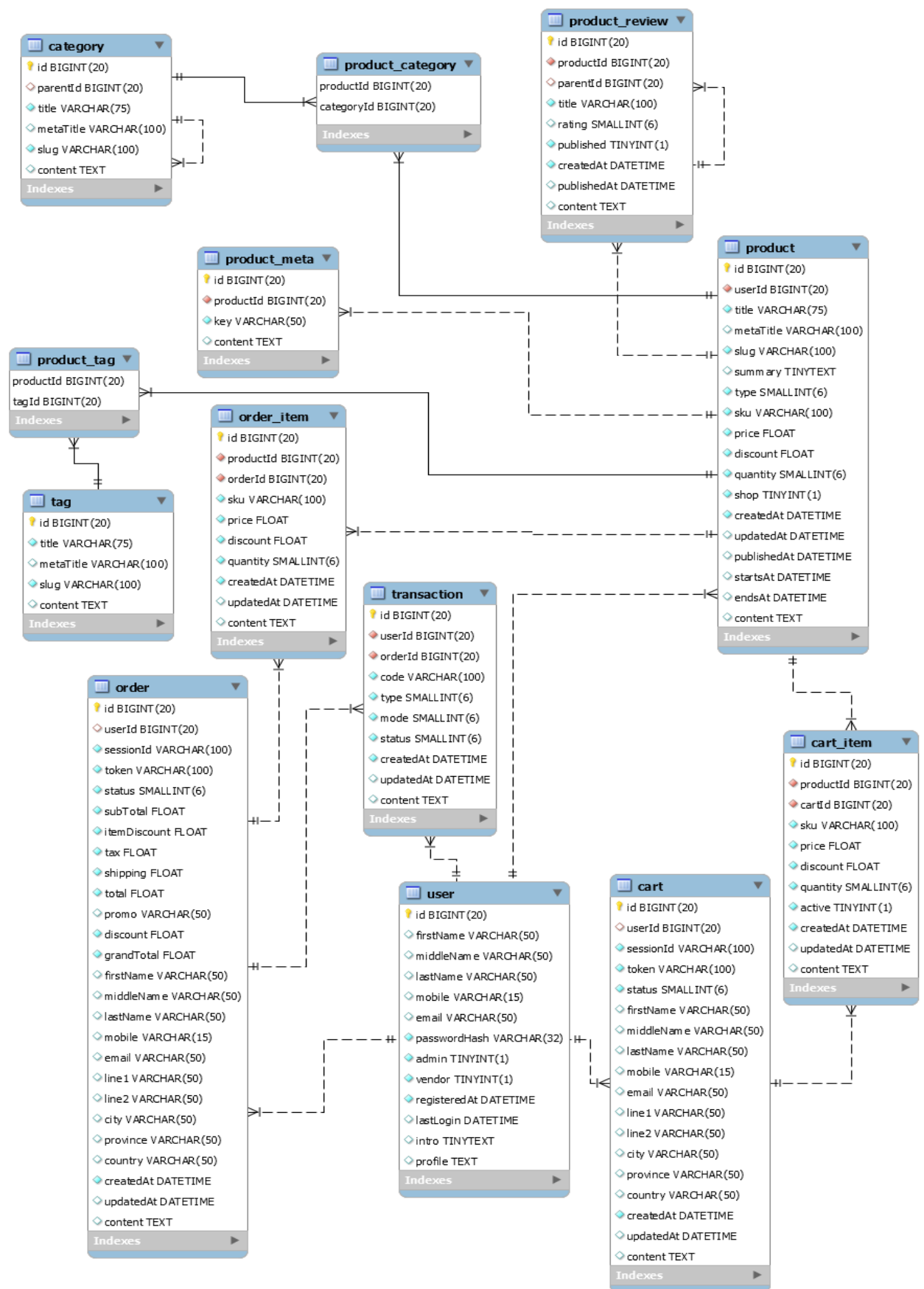


Рисунок 2.9 СУБД приложения

2.4 Проектирование пользовательского интерфейса

Интерфейс - в широком смысле определенная стандартами граница между взаимодействующими независимыми объектами. Интерфейс задает параметры, процедуры и характеристики взаимодействия объектов.

Интерфейс пользователя - элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением.

Пользовательский интерфейс (ПИ) программного средства реализован в соответствии с требованиями к пользовательскому интерфейсу, определенными в процессе анализа требований. Разработан состав, иерархия диалоговых страниц, т.е. логика работы пользовательского интерфейса.

Разработка проводилась на языке PHP. Далее приводится описание страниц.

С точки зрения пользователя при наборе в браузере URL <http://localhost:3000/> появляется главное окно сайта, на котором присутствуют элементы меню (рисунок 2.10).

При нажатии на элемент меню галерея пользователь попадает в галерею товара (рисунок 2.12). Далее пользователь с помощью меню выбирает или категории товаров и/или конкретный товар.

При выборе конкретного товара попадает на страницу с характеристиками товара, и можем произвести занесение товара в корзину. После чего иконка корзины меняется и показывает количество товара в корзине. Пользователь также может отменить заказ. Если же заказ подтвержден, пользователь переходит на страницу оплаты. После чего он может вернуться на страницу выбора товара.

КОЛЛЕКЦИИ ПОД ВАШИМ БРЕНДОМ

От идеи до выхода на подиум — 30 дней

ПОДРОБНЕЕ —

01

02

03

04

05

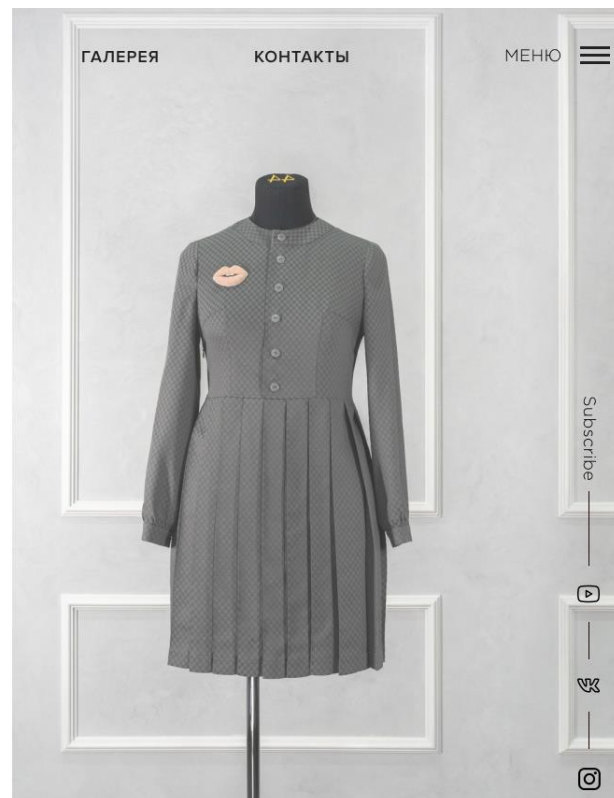


Рисунок 2.10 Главное окно

ОСТАВЬТЕ ЗАЯВКУ

КАК ВАС ЗОВУТ?

АДРЕС ЭЛЕКТРОННОЙ ПОЧТЫ

ПАРУ СЛОВ О ЗАКАЗЕ

ОТПРАВИТЬ —



Рисунок 2.11 Оставить заявку

					ПГУ 2.090301.36.001 ПЗ	Лист
						57
Изм.	Лист	№ докум	Подпись	Дата		

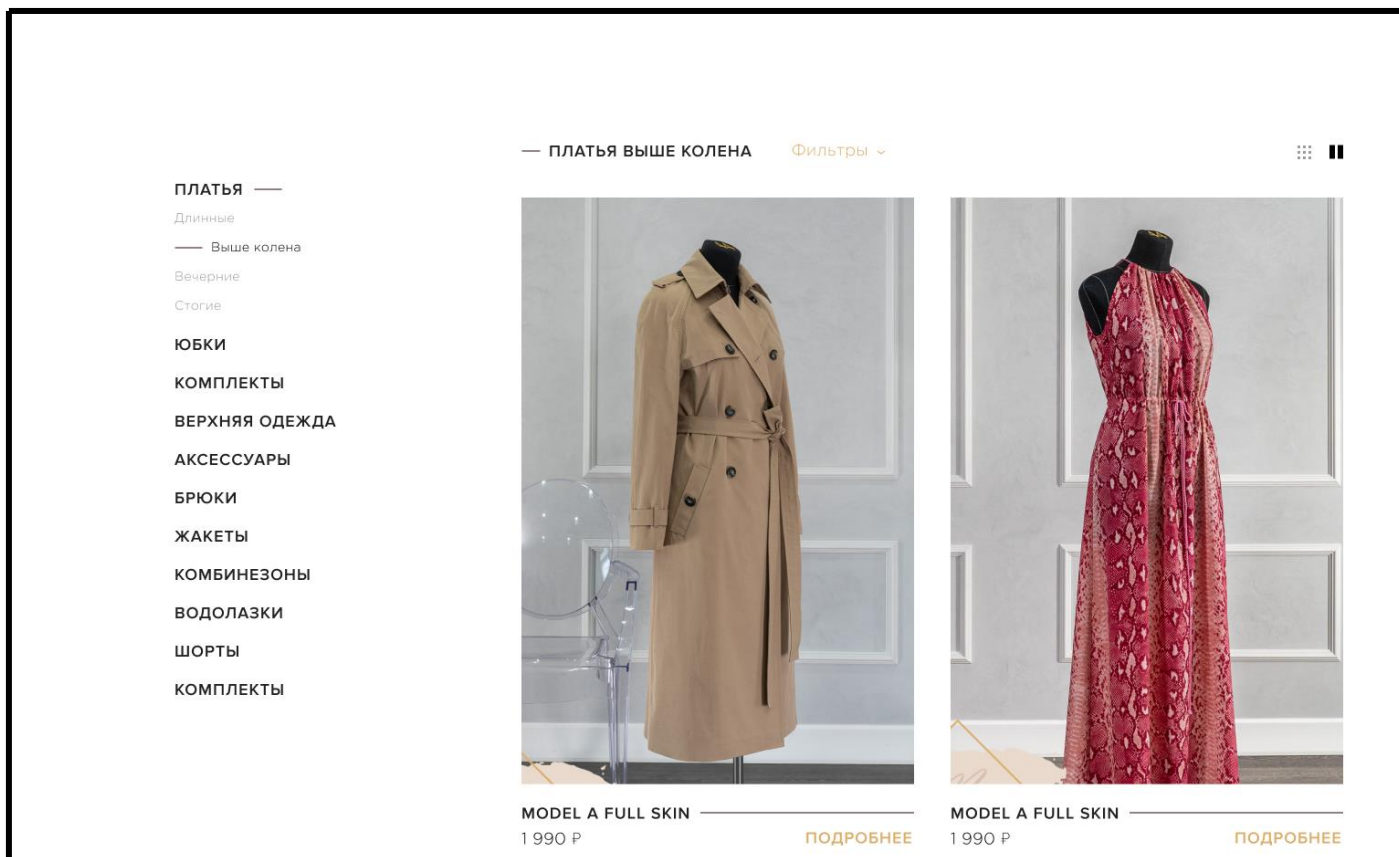


Рисунок 2.12 Основная галерея

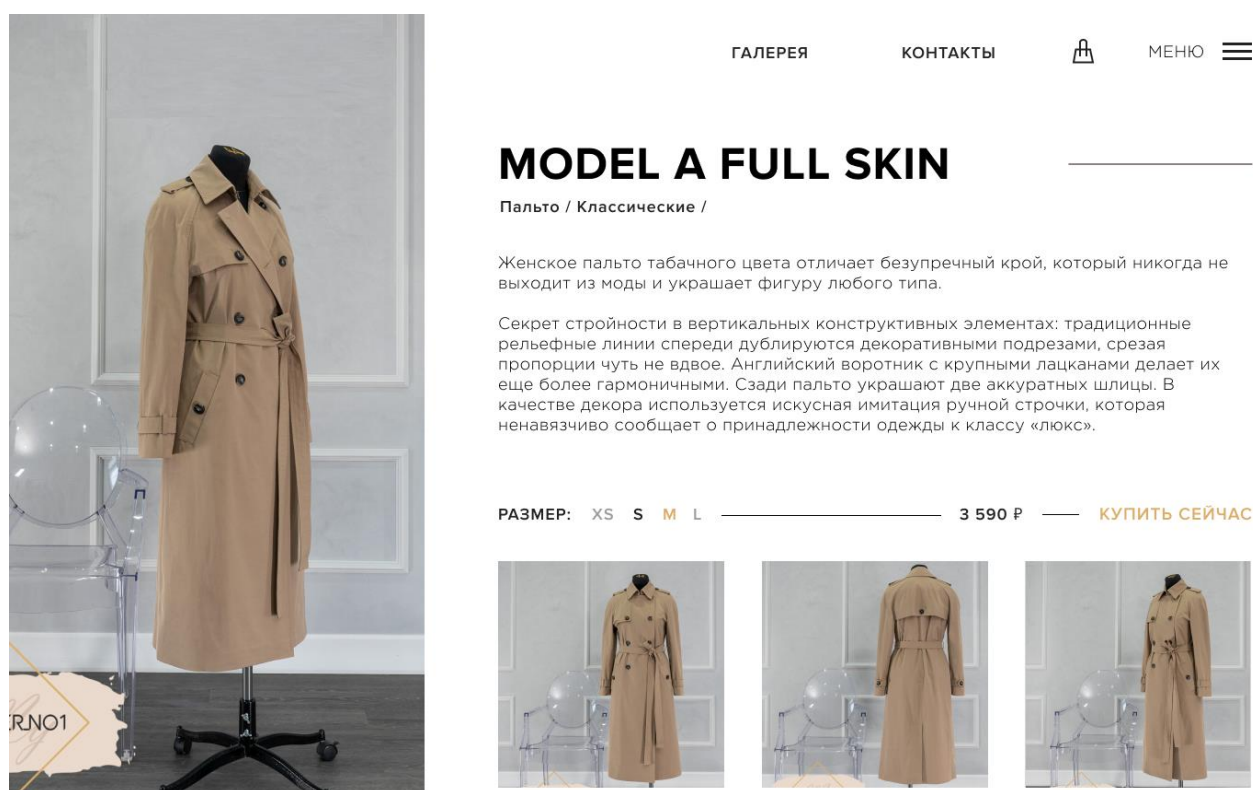


Рисунок 2.13 Выбор товара

ГАЛЕРЕЯ КОНТАКТЫ  МЕНЮ 

КОРЗИНА

ПРОДУКТ	ЦВЕТ	РАЗМЕР	КОЛИЧЕСТВО	ЦЕНА	
	Model A full skin	Красный	XS	- 1 +	1 000 Р УДАЛИТЬ

СУММА К ОПЛАТЕ: 1 000 Р

[ОФОРМИТЬ ЗАКАЗ](#) —

Рисунок 2.14 Корзина заказов

ГАЛЕРЕЯ КОНТАКТЫ  МЕНЮ 

ОФОРМЛЕНИЕ ЗАКАЗА

СПОСОБ ДОСТАВКИ



СУММА К ОПЛАТЕ: 1 700 Р

[ОПЛАТИТЬ](#) —

Выберетеспособ доставки чтобы
оплатить

Рисунок 2.15 Оформление заказа

Андрей

tim@gmail.com

КОММЕНТАРИЙ К ЗАКАЗУ

ВЫБЕРИТЕ СПОСОБ ДОСТАВКИ



САМОВЫВОЗ

ПО МОСКВЕ

+300 ₽

ПО РОССИИ

+450 ₽

Рисунок 2.16 Способ доставки



ОФОРМЛЕНИЕ ЗАКАЗА

1 000 ₽

Ship to

John Doe
Russia, Moscow, Tverskaya 18

☐ Make this my preferred shipping address

Change >

Pay with

☒ PayPal Balance

PREFERRED

☐ MasterCard x-2114

+ Add a debit or credit card

Manage >

Pay Now

PayPal is the safer, faster way to pay

No matter where you shop, we keep your financial information secure.

[Cancel and return to Gearbest](#)[Legal](#) [User Agreement](#) [Privacy](#) [Feedback](#) © 1999 - 2019

Рисунок 2.17 Платеж PayPal



ОФОРМЛЕНИЕ ЗАКАЗА

Ваш заказ №311230007 успешно оформлен и оплачен, вы получите уведомление когд можно будет его забрать!

[ПРОДОЛЖИТЬ ПОКУПКИ](#) —

Рисунок 2.18 Заказ оплачен

					ПГУ 2.090301.36.001 ПЗ	Лист
						61
Изм.	Лист	№ докум	Подпись	Дата		

2.6 Интеграция платежной системы в структуру приложения

В данном разделе показаны способы интеграция платежей, а также оплаты по подписке с помощью PayPal в РНР.

Для реализации данной схемы нам потребуется business аккаунт. PayPal Payments Standard должно быть достаточно.

Переходим по ссылке и создаем аккаунт [url https://www.paypal.com/welcome/signup/#/email_password].

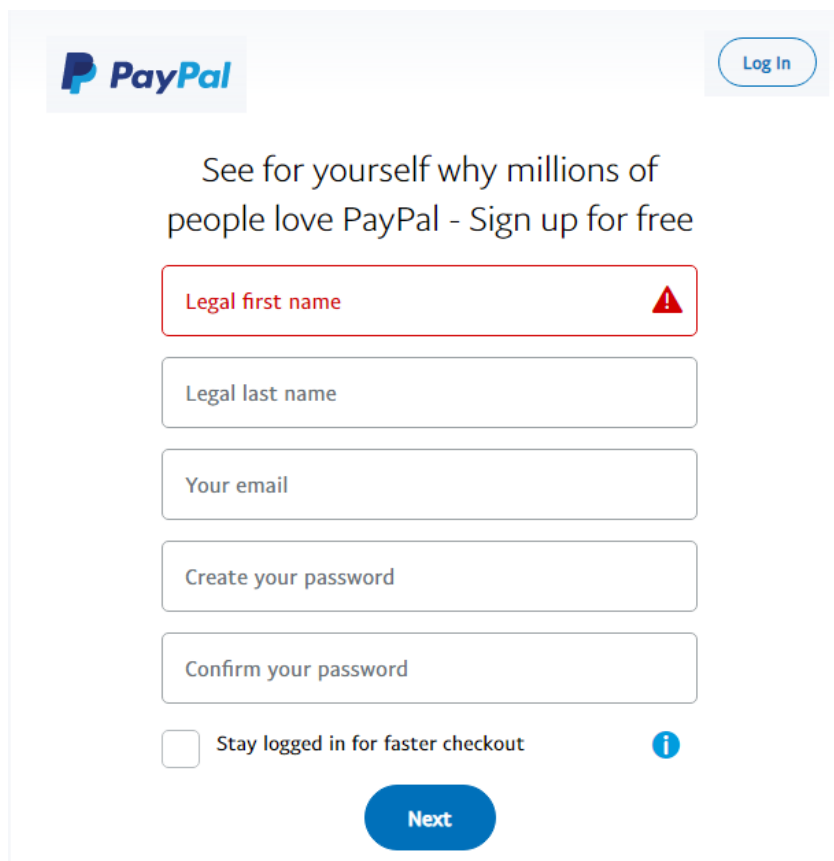


Рисунок 2.19 Вход в систему PayPal

Для тестирования приложения будет использоваться PayPal Sandbox. Потребуется два sandbox аккаунта. Аккаунт покупателя(buyer) и аккаунт продавца(facilitator). Прежде всего нужно задать пароль для обоих sandbox аккаунтов. Для этого переходим на сайт paypal в раздел для разработчиков. Логинимся, затем переходим в dashboard. В меню слева находим раздел Sandbox, вкладку accounts. Здесь пользователь увидит 2 sandbox аккаунта (Buyer и Facilitator).

					ПГУ 2.090301.36.001 ПЗ	Лист
						62
Изм.	Лист	№ докум	Подпись	Дата		

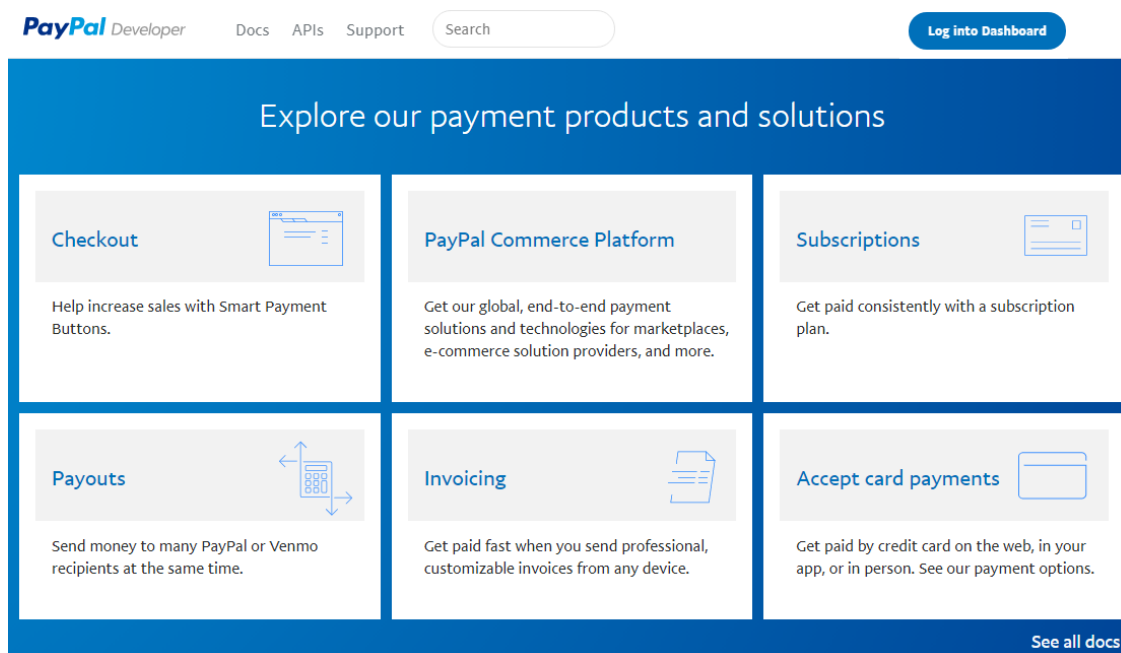


Рисунок 2.20 Выбор операции в системе PayPal

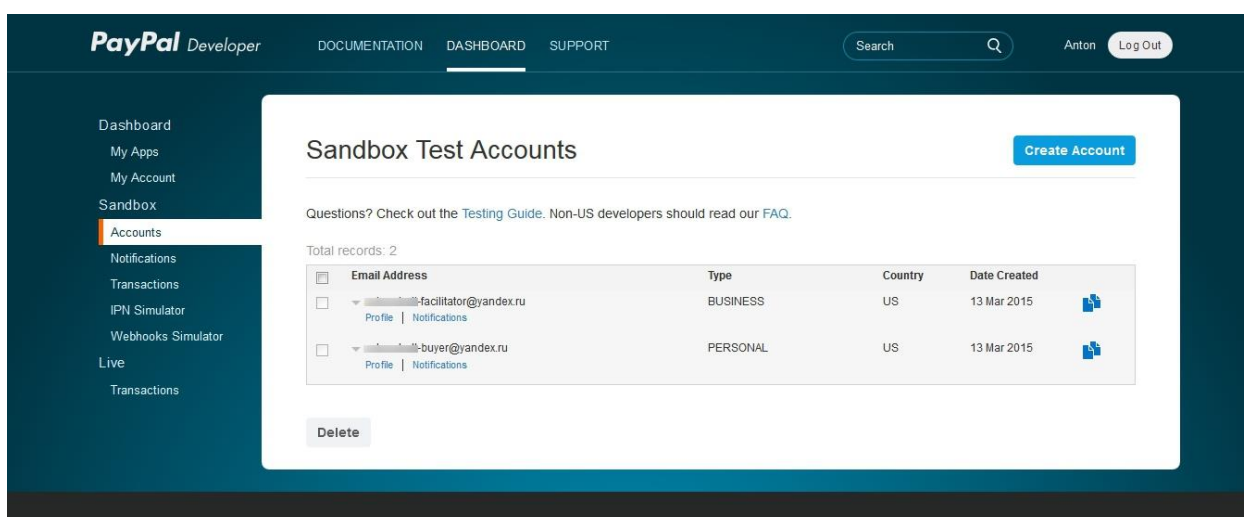


Рисунок 2.21 Шаги PayPal по настройке платежей в третьей системе

Нажимаем на profile, в появившемся модальном окне кликаем change password, затем сохраняем пароль.

Устанавливаем пароли для обоих аккаунтов. После этого можно перейти на сайт Paypal Sandbox и попробовать залогиниться.

Теперь нужно настроить Paypal Facilitator аккаунт, на который будем получать средства. Переходим на сайт Sandbox, логинимся с помощью

facilitator аккаунта и переходим в настройки профиля. Открываем меню profile, выбираем пункт my selling tools.

В разделе Selling online выбираем пункт Website preferences, нажимаем Update. Здесь можно включить перенаправление пользователя. После завершения платежа пользователь по умолчанию будет перенаправлен на указанный URL. Но также есть возможность перенаправить пользователя на другой URL.

Test Site

Log Out | Security and Protection

PayPal

My Account | Send Money | Request Money | Merchant Services | Products & Services

Overview | Add Money | Withdraw | History | Resolution Center | Profile

Website Payment Preferences [Back to My Profile](#)

Auto Return for Website Payments

Auto Return for Website Payments brings your buyers back to your website immediately after payment completion. Auto Return applies to PayPal Website Payments, including Buy Now, Donations, Subscriptions, and Shopping Cart. [Learn More](#)

Auto Return: ☒ On ☐ Off

Return URL: Enter the URL that will be used to redirect your customers upon payment completion. This URL must meet the guidelines detailed below. [Learn More](#)

Return URL:

Return URL Requirements: The following items are required in order to set up Auto Return.

- Per the user agreement, you must provide verbiage on the page displayed by the Return URL that will help the buyer understand that the payment has been made and that the transaction has been completed.
- You must provide verbiage on the page displayed by the Return URL that explains that payment transaction details will be emailed to the buyer.

Рисунок 2.22 Шаги PayPal по настройке платежей в третьей системе

Также необходимо активировать Paypal Instant Payment Notifications. Для этого в разделе Getting paid and managing my risk выбираем пункт Instant payment notifications и также нажимаем Update.

My business info > Email, address, phone, password, more.

My money > Bank accounts, cards, more.

My settings > Notifications, customer ID, more.

My selling tools > Getting paid, shipping, more.

Quick links

- [Update Email](#)
- [Update Password](#)
- [Update Bank Account](#)
- [Update Card](#)
- [Update Street Address](#)
- [Update Phone Number](#)
- [Add Passport Number](#)
- [Add Permanent Resident/Green Card Number](#)

My business setup Cool tools to help boost my business. [Get started](#)

☒ **Selling online**

PayPal buttons	Manage my payment buttons.	Update
Credit card statement name	Name of my business as it appears on customer card statements: TESTFACILIT	Update
Sales tax	Set up sales taxes for multiple regions.	Update
Custom payment pages	Set up PayPal payment pages to look like my website.	Update
Website preferences	Bring customers back to my website after they pay with PayPal.	Update
API access	Manage API credentials to integrate my PayPal account with my online store or shopping cart.	Update
Invoice templates	Create and manage my invoices.	Update

☒ **Getting paid and managing my risk**

My automatic payments	Manage the subscriptions, automatic payments, and installment plans that I offer my customers.	Update
Instant payment notifications	Integrate PayPal payment notifications with my website.	Update
Block payments	Limit payments, add instructions, and more.	Update
Customer service message	Create a personalized message for customer disputes.	Update

☒ **Shipping my items**

https://www.sandbox.paypal.com/cgi-bin/customerprofileweb?cmd=_profile-ipn-notify

Рисунок 2.23 Шаги визарда PayPal по настройке платежей в третьей системе

В настройках IPN указываем URL, на котором будет работать наш IPN Listener. Этот URL обязательно должен быть доступен глобально т.к. на него будут приходить уведомления о проведении операций.

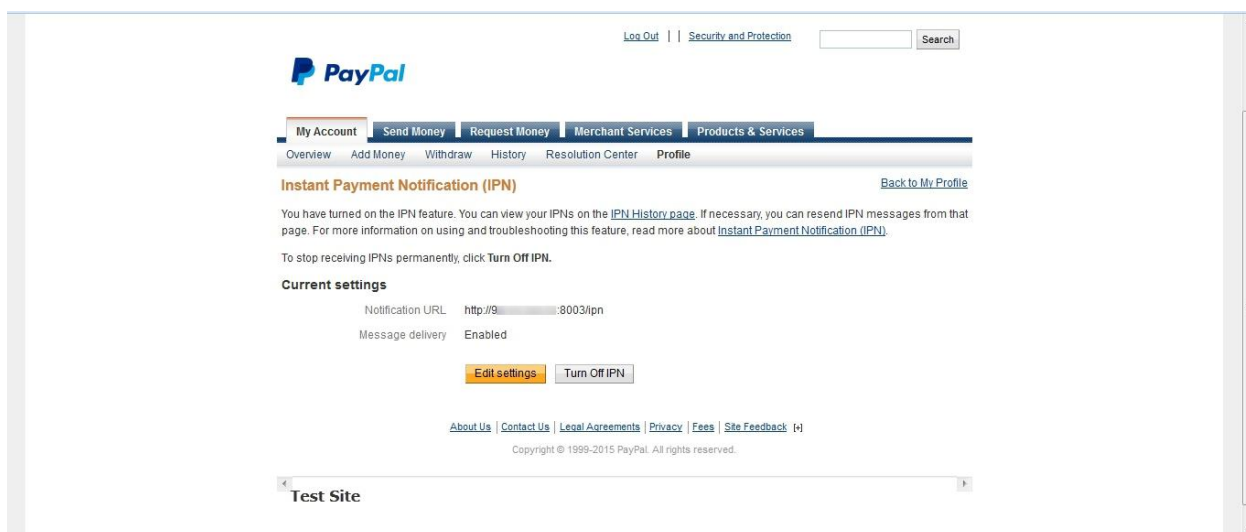


Рисунок 2.24 Шаги визарда PayPal по настройке платежей в третьей системе

Включаем опцию Message delivery и сохраняемся. На этом настройка аккаунта завершена. Можно приступить к настройке непосредственно платежей.

2.6.1 Разовые Платежи

Для начала реализуем разовые платежи. Это, вероятно, наиболее распространенный вариант использования. Пользователь просто хочет купить какой-нибудь товар. Это позволит не менять что либо в настройках PayPal.

Список товаров и цены хранятся в базе нашего приложения, программисты могут их менять, как требуется. Для разовых платежей будем использовать Payment Buttons (PayPal Payments Standard).

Список товаров храниться в базе данных нашего приложения. У программистов есть возможность добавлять удалять и редактировать товары в любой момент. Здесь представлена наиболее простая структура, вся информация храниться в одной таблице.

Так же имеется возможность менять цену в зависимости от количества заказанных товаров, или менять стоимость в зависимости от дня недели и времени.

Или включить в заказ много различных товаров. Таблица products — здесь будем хранить товары, users — здесь будем хранить пользователей, products_users — кому мы и что отгрузили, в таблице transactions - будем хранить в нашей базе историю транзакций.

Для формирования формы оплаты, для начала создадим форму заказа. Генерируем форму в нашем приложении, где указываем основные параметры заказа(название товара, цена, количество).

Здесь мы можем указать любую цену, название, количество и т.д. Поле custom полезно тем, что в нем можно передавать любые данные. Здесь мы будем передавать id товара, id пользователя и, возможно, другую информацию. Эти данные понадобятся нам для дальнейшей обработки платежа.

Если нужно передать несколько параметров, можно использовать json или сериализацию. Либо можно использовать дополнительные поля вида on0, on1, os0 and os1.

Ниже приведен код формы:

```
<?php $payNowButtonUrl = 'https://www.timandr.paypal.com/cgi-bin/websc';
$userId = 3155689755543
$receiverEmail = 'andrey-facilitator@gmail.com'; $productId = 1; $itemName =
'Product 1';
$amount = '1.0';
$quantity = 3;
$returnUrl = 'http://your-site.com/single_payment?status=paymentSuccess';
$customData = ['user_id' => $userId, 'product_id' => $productId]; ?>
<form action="<?php echo $payNowButtonUrl; ?>"
method="post"> <input type="hidden" name="cmd" value="_xclick">
    <input type="hidden" name="business" value="<?php echo $receiverEmail;
    ?>">
    <input id="paypalItemName" type="hidden" name="item_name" value="<?php
    echo $itemName; ?>">
```

```

<input id="paypalQuantity" type="hidden" name="quantity" value="<?php echo
$quantity; ?>">
<input id="paypalAmmount" type="hidden" name="amount" value="<?php echo
$amount; ?>">
<input type="hidden" name="no_shipping" value="1">
<input type="hidden" name="return" value="<?php echo $returnUrl; ?>">
<input type="hidden" name="custom" value="<?php echo
json_encode($customData);?>">
<input type="hidden" name="currency_code" value="USD">
<input type="hidden" name="lc" value="US">
<input type="hidden" name="bn" value="PP-BuyNowBF"> <button type="submit"> Pay
Now </button>
</form>

```

Здесь пользователь может оплатить заказ с помощью PayPal аккаунта либо с помощью банковской карты. Далее пользователь переадресовывается обратно на наш сайт (параметр return), где мы можем ему сообщить, что его платеж находится в обработке.

Рисунок 2.25 Указание почтового ящика для отправки платежей

После того, как пользователь совершил платеж, PayPal обрабатывает его и отправляет подтверждение в наше приложение. Для этого используется сервис Instant Payment Notification(IPN).

В начале процесса настройки PayPal аккаунта требуется установить модуль IPN Notification URL. Сейчас самое время создать IPN listener, который будет обрабатывать IPN запросы. PayPal предоставляет пример реализации IPN listener. В двух словах, как это работает: PayPal обрабатывает платеж пользователя, видит что все хорошо и платеж успешно завершен. После этого IPN отправляет на наш Notification URL такого вида Post запрос:

```
mc_gross=37.50&protection_eligibility=Ineligible&payer_id=J86MHHMUDEHZU&tax=0.00&
payment_date=07%3A04%3A48+Mar+30%2C+2015+PDT&payment_status=Completed&charset=win
dows-
1252&first_name=test&mc_fee=1.39-ify_version=3.8&custom=%7B%22user_id%22%3A314%2C
%22service_provider%22%3A%22twilio%22%2C%22service_name%22%3A%22textMessages%22%7
D&payer_status=verified&business=antonshel-
facilitator%40gmail.com&quantity=150&verify_sign=AR-ITpb83c-
ktcbmApqG4jM170eQAx2RSvfYZo4XU8YFZrTSeF.iYsSx&payer_email=antonshel-
buyer%40gmail.com&txn_id=30R69966SH780054J&payment_type=instant&last_name=buyer&r
eceiver_email=antonshel-
facilitator%40gmail.com&payment_fee=1.39&receiver_id=VM2QHCE6FBR3N&txn_type=web_a
ccept&item_name=GetScorecard+Text+Messages&mc_currency=USD&item_number=&residence
_country=US&test_ipn=1&handling_amount=0.00&transaction_subject=%7B%22user_id%22%
3A314%2C%22service_provider%22%3A%22twilio%22%2C%22service_name%22%3A%22textMessa
ges%22%7D&payment_gross=37.50&shipping=0.00&ipn_track_id=6b01a2c76197
```

Наш IPN Listener должен этот запрос обработать. В частности:

- Проверить тип запроса(разовый платеж либо подписка). В зависимости от этого по-разному будем его обрабатывать. В нашем случае это будет разовый платеж — web_аccept.
- Выбрать окружение — sandbox либо live.
- Проверить достоверность запроса. Зная как выглядит IPN запрос и, зная наш IPN Notification URL, любой желающий может отправить нам поддельный запрос. Поэтому мы обязательно должны выполнить эту проверку.

```
class PayPalIpn {
    private $debug = true;
    private $service;
    public function createIpnListener(){
```

```

        $postData = file_get_contents('php://input');
        $transactionType = $this->getPaymentType($postData);
        $config = Config::get();
        if($transactionType ==
        PaypalTransactionType::TRANSACTION_TYPE_SINGLE_PAY){ $this-
        >service = new PaypalSinglePayment();
    } elseif($transactionType ==
        PaypalTransactionType::TRANSACTION_TYPE_SUBSCRIPTION){
        $this->service = new PaypalSubscription($config);
    } else{
        throw new Exception('Wrong payment type');
    }
    $raw_post_data = file_get_contents('php://input');
    $raw_post_array = explode('&', $raw_post_data);
    $myPost = array();
    foreach ($raw_post_array as $keyval) {
        $keyval = explode ('=', $keyval);
        if (count($keyval) == 2)
            $myPost[$keyval[0]] = urldecode($keyval[1]);
    }
    $customData = $customData = json_decode($myPost['custom'],true);
    $userId = $customData['user_id'];
    $req = 'cmd=_notify-validate';
    if(function_exists('get_magic_quotes_gpc')) {
        $get_magic_quotes_exists = true;
    } else{
        $get_magic_quotes_exists = false;
    }
}

```

После этого, если PayPal подтвердил подлинность запроса, можем приступить к его дальнейшей обработке.

2.6.2 Обработка платежа

В первую очередь потребуется получить значение поля custom, где система передавала id заказа, id пользователя или еще что-то (зависит от логики нашего приложения). Соответственно пользователи смогут получить из нашей базы данных информацию о пользователе/заказе. Также нужно получить id транзакции.

Paypal может несколько раз присылать подтверждение одной и той же транзакции. Поэтому нужно проверить и, если транзакция не

					ПГУ 2.090301.36.001 ПЗ	Лист
						69
Изм.	Лист	№ докум	Подпись	Дата		

обрабатывалась, обрабатываем ее. Если транзакция уже обрабатывалась, тогда ничего не делаем.

Проводим валидацию платежа. Если все нормально, тогда можно сохранить информацию о платеже в базу и выполнить дальнейшие действия (присвоить пользователю статус «premium», заказу статус «оплачен» и т.д.). Если платеж не прошел валидацию, необходимо установить причину и связаться с пользователем. Дальнейшие операции, в частности, отмена платежа, проводятся вручную.

2.6.3 Валидация платежа

Валидация платежа сильно зависит от бизнес-логики вашего приложения. Могут быть добавлены специфические условия. Например пользователь оплатил 15 единиц товара, а в наличии есть всего 10. Нельзя пропустить такой заказ.

Впрочем, такие вещи имеет смысл проверять еще на этапе генерации формы. Валидация платежа нужна скорее для предотвращения мошенничества(например пользователь в форме оплаты вручную увеличил количество товара, а цену оставил прежней).

```

<?php
    function processPayment($myPost){

        $customData = json_decode($myPost['custom'],true);
        $userId = $customData['user_id'];
        $productId = $customData['product_id'];

        //
        $userService = new UserService();
        $userInfo = $userService->getUserData($userId);

        //получаем информацию о транзакции из базы данных
        $transactionService = new TransactionService();
        $transaction = $transactionService->getTransactionById($myPost['txn_id']);

        if($transaction === null){
            //получаем информацию о продукте из бд
            $productService = new ProductService();
            $product = $productService->getProductById($productId);

            // проводим валидацию транзакции
            if($this->validateTransaction($myPost,$product)){
                // оплата прошла успешно. сохраняем транзакцию в базу данных.
                $transactionService->createTransaction($myPost);

                // Выполняем какие-либо другие действия
            }

            else{
                // платеж не прошел валидацию. Необходимо проверить вручную
            }
        }
        else{
            //дубликат, эту транзакцию мы уже обработали. ничего не делаем
        }
    }
?>

```

Есть несколько вещей, которые в любом случае стоит проверить:

- Проверить соответствие цены в платеже и в нашей базе данных
- Проверить что итоговая стоимость не равна 0(паранойя т.к. предыдущий пункт покрывает этот случай)
- Проверить, что указан правильный получатель платежа
- Проверить статус платежа
- Проверить валюту платежа

```

<?php
function validateTransaction($myPost,$product){
    $valid = true;

    /*
     * Проверка соответствия цен
     */
    if($product->getTotalPrice($myPost['quantity']) != $myPost['payment_gross']){
        $valid = false;
    }

    /*
     * Проверка на нулевую цену
     */
    elseif($myPost['payment_gross'] == 0){
        $valid = false;
    }

    /*
     * Проверка статуса платежа
     */
    elseif($myPost['payment_status'] != 'Completed'){
        $valid = false;
    }

    /*
     * Проверка получателя платежа
     */
    elseif($myPost['receiver_email'] != 'YOUR PAYPAL ACCOUNT'){
        $valid = false;
    }

    /*
     * Проверка валюты
     */
    elseif($myPost['mc_currency'] != 'USD'){
        $valid = false;
    }
}

```

После внедрения данного кода на странице оплата приложения появится элемент оплаты (рисунок 2.17).

3. ТЕХНОЛОГИЯ РАЗРАБОТКИ И ОТЛАДКИ WEB-ПРИЛОЖЕНИЙ В СРЕДЕ NETBEANS

Для реализации данного дипломного проекта была выбрана среда разработки NetBeans.

Эта среда — бесплатная, быстрая, кросс-платформенная, хорошо понимает язык PHP. В ней приятно писать PHP-код, потому что присутствует все необходимые для этого инструменты — подсветка, completion из всех подключенных файлов, браузер классов PHP, всплывающая документация по стандартным функциям и так далее. Вместе с тем, это не просто текстовый редактор. NetBeans умеет скачивать и закачивать файлы на удаленные сервера по FTP и SFTP, имеет развитый SQL-терминал и полностью автоматизирует работу с SVN или CVS.

3.1 Создание проекта в NetBeans

Ниже описана технология создания проекта в среде Eclipse.

Первый шаг, открытие помощника создания проекта - **File > New > Project > PHP > PHP Application**.

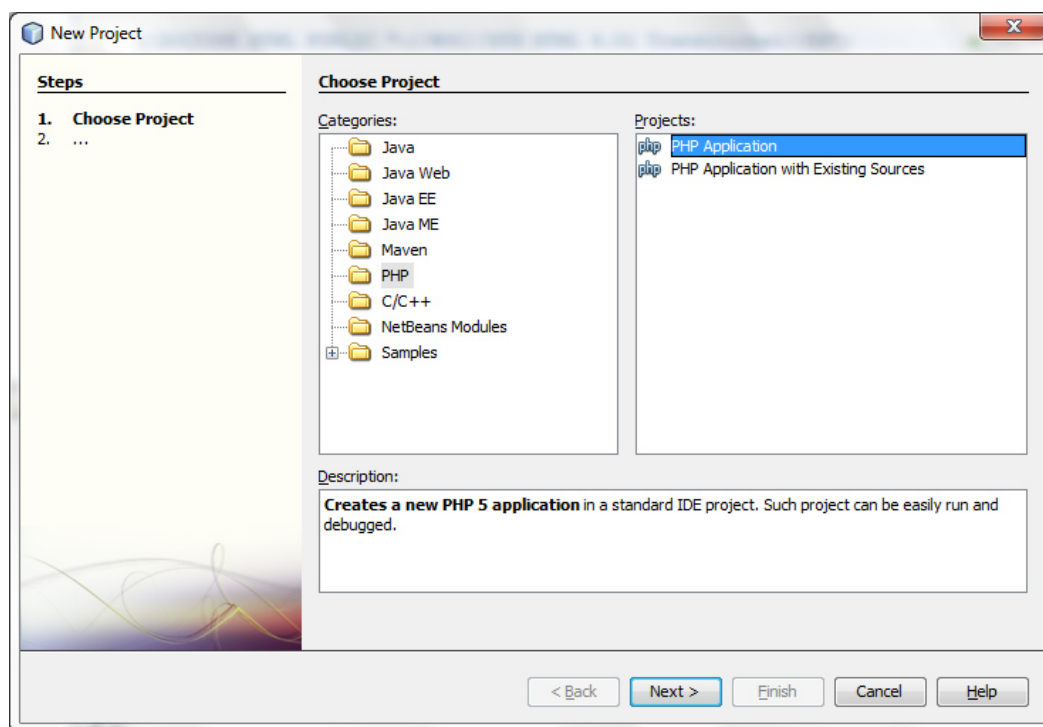


Рисунок 3.1 – Создание нового проекта

Нажмите кнопку **Next** и перейдите в диалог настройки пространства имен проекта (рисунок 3.2).

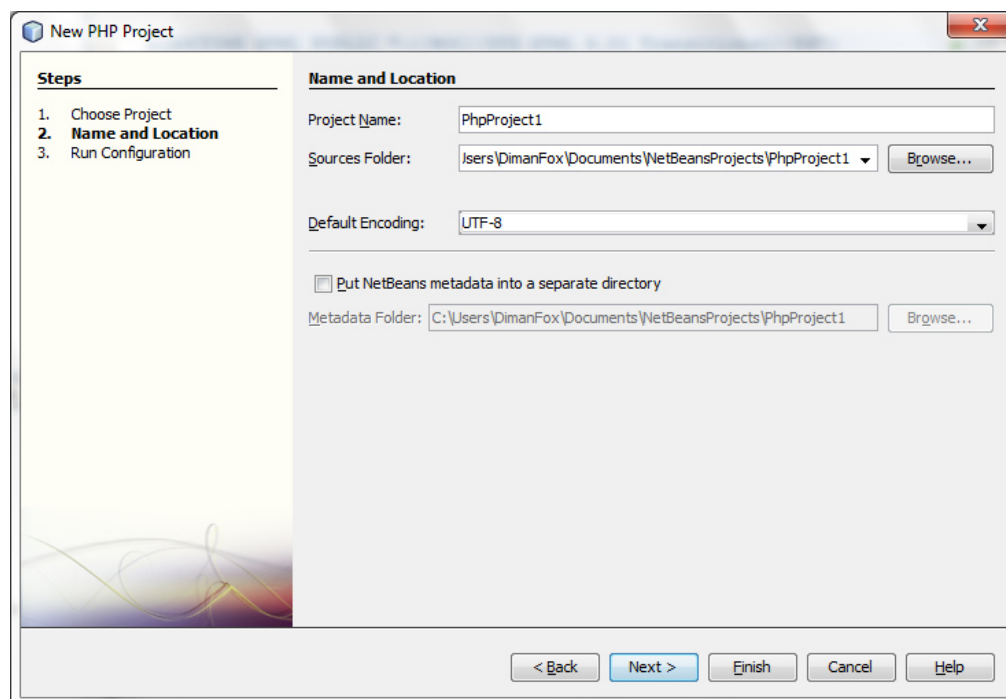


Рисунок 3.2 – Настройка пространства имен проекта

Нажмите кнопку **Next** и перейдите в диалог конфигурации проекта (рисунок 3.3).

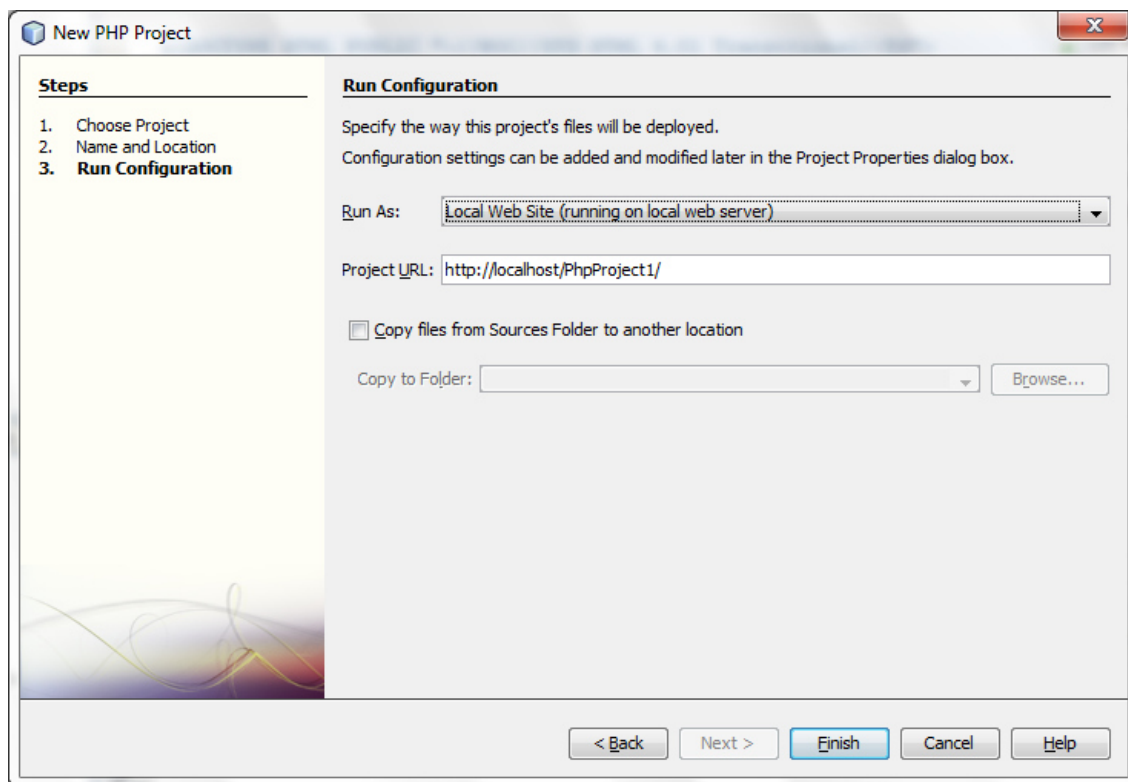


Рисунок 3.3 – Настройка конфигурации проекта

В результате вышеперечисленных операций появляется окно нового проекта, в котором присутствует структура каталога проекта и сгенерированные автоматически файлы настройки (рис. 3.4).

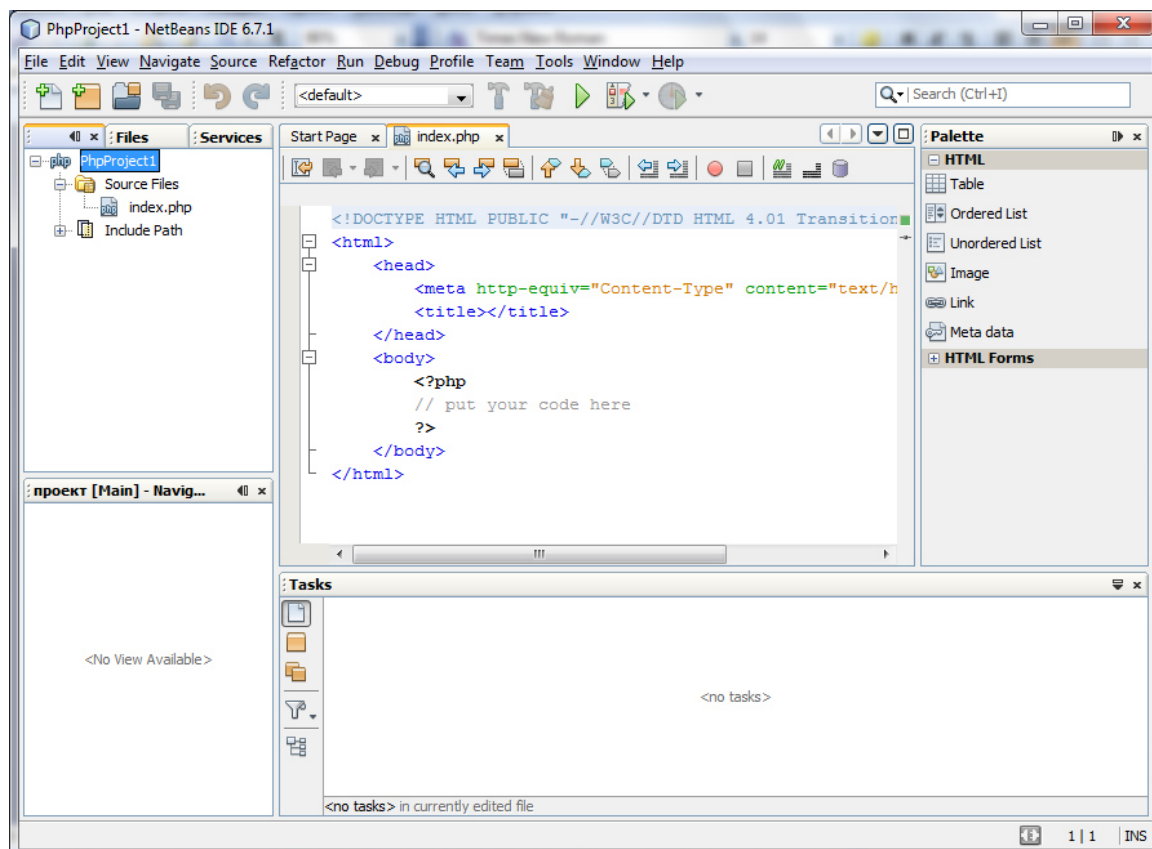


Рисунок 3.4 – Окно нового проекта

По умолчанию в проект добавлен файл `index.php`, который является главным файлом проекта, и к которому будут подключаться остальные файлы.

Для добавления новых файлов в проект нужно воспользоваться пунктом меню - **File > New > New File**.

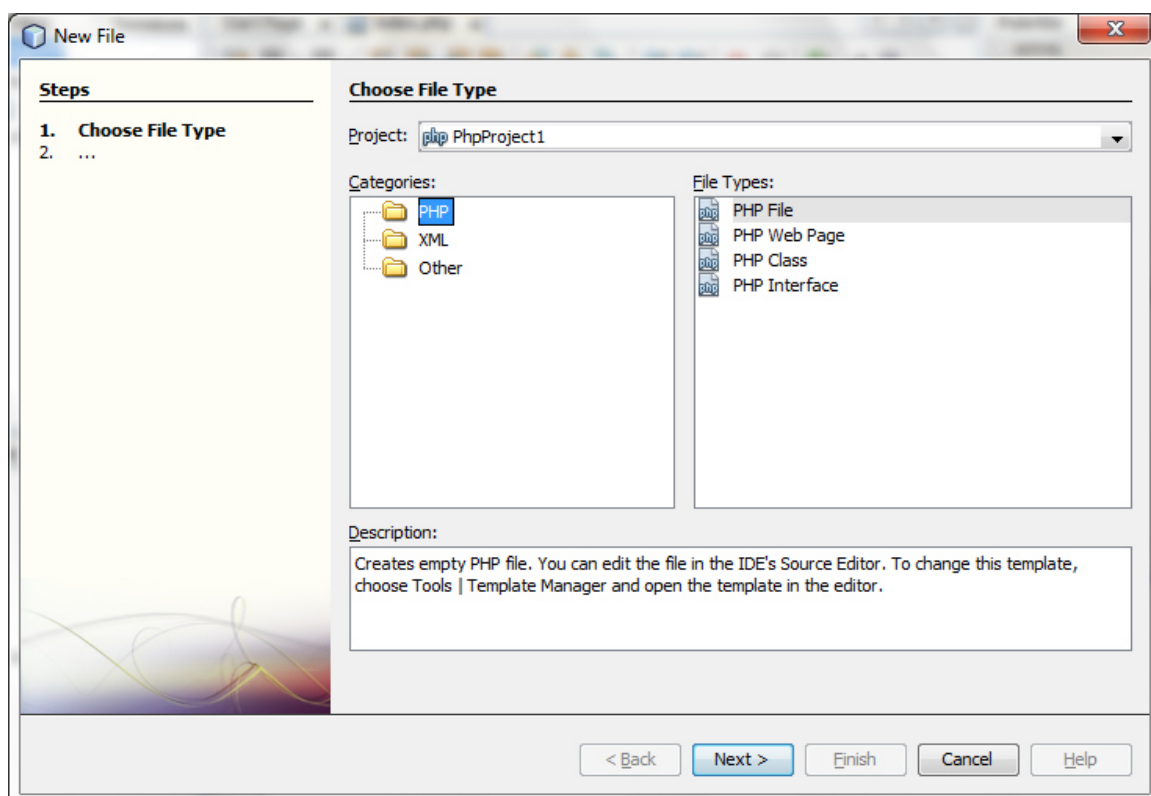


Рисунок 3.5 – Добавление нового файла в проект

После того как выбрали тип файла, который нужно добавить в проект, следует нажать кнопку «Next» для того чтобы задать имя файлу.

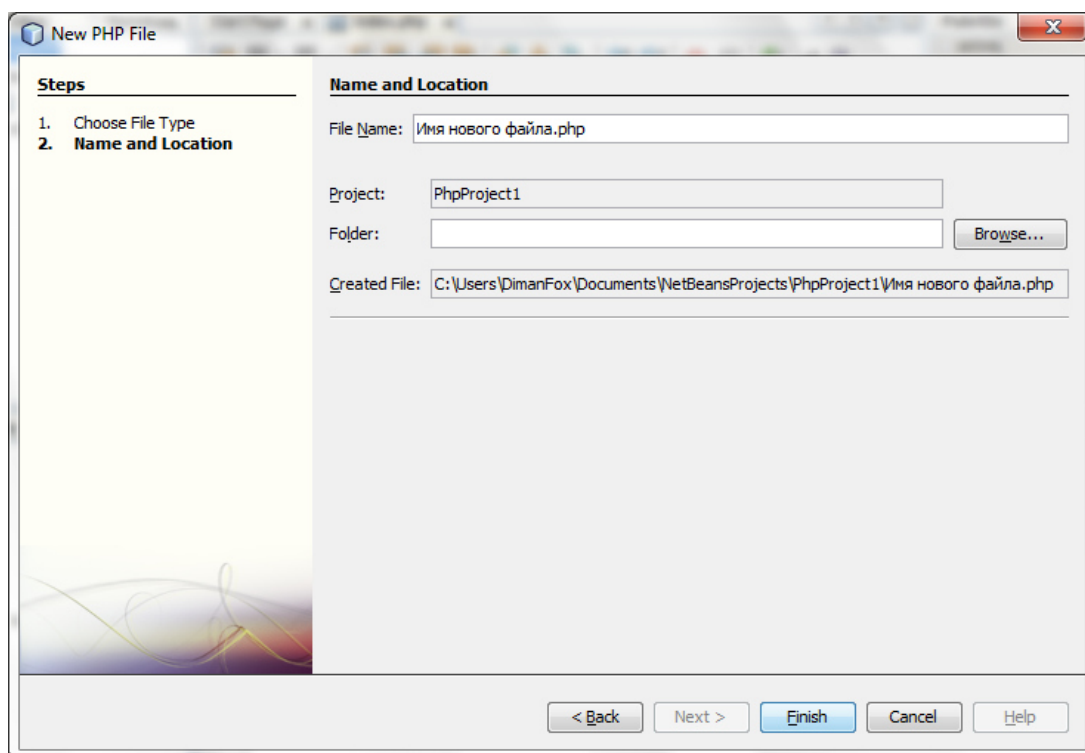


Рисунок 3.6 – Добавление нового файла в проект

После того как задали имя новому файлу, можно нажать кнопку «Finish» и новый файл добавиться в проект с минимум кода-шаблоном.

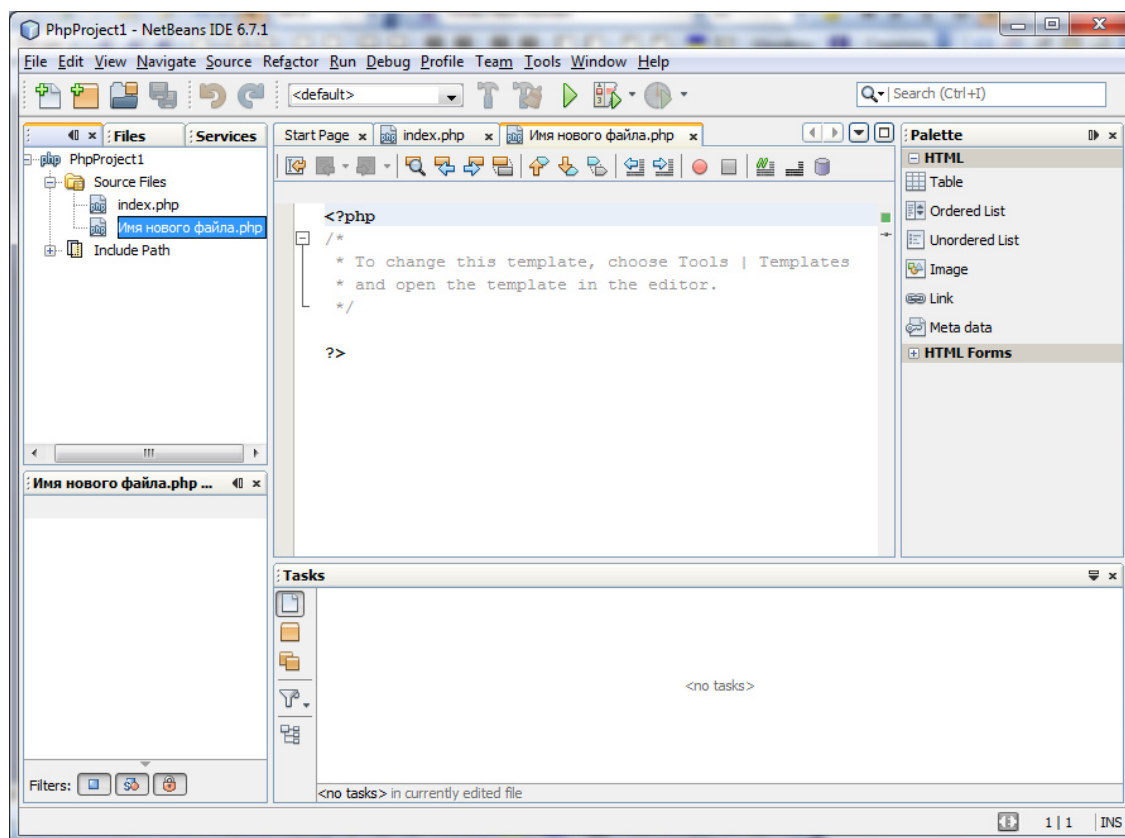


Рисунок 3.7 – Окно проекта

3.2 Отладка кода с помощью XDebug

Отладка программного обеспечения самая ненавистная работа для разработчика. Большинство используют для отладки связку `echo(print_r,var_dump)` и `exit(die)`, переходя от одной строке к другой. Однако, если ошибка появляется вновь в этом файле, требуется заново прописывать отладочные команды. Можно конечно отладку обернуть внутри конструкции `if`, которая будет работать только, когда определена какая-либо константа, например `DEBUG`. Но каждый такой `if` будет слегка замедлять производительность, да и смотреться будет в коде очень ужасно.

Xdebug позволяет создавать логи трассировки, достаточно хороший выход для данной ситуации, и не надо будет изменять программу. Однако лог трассировки, особенно когда он создан для части программы, предоставляет нам много информации, не имеющей никакого отношения к отладке, таким образом, использование отладчика это оптимальное решение. Отладчик позволит вам остановить запуск программы на некоторое время, проверить или модифицировать текущее значение переменной, а затем продолжить выполнение программы. Запуская программу шаг за шагом, вы можете близко взглянуть как выполняется ваш код, что поможет найти вам места ошибок.

В среде разработки NetBeans IDE для отладки PHP кода есть очень удобное решение – Xdebug. Для его настройки необходимо:

В соответствии с версией php, выбрать и скачать библиотеку xdebug .

Скопировать файл php_xdebug.dll в папку /php/ext/.

В файле php.ini добавить секцию XDebug и прописать в ней следующее:

```
[XDebug]
```

```
extension=php_xdebug-2.0.5-5.2.dll
```

```
zend_extension_ts="C:\server\php\ext\php_xdebug-2.0.5-5.2.dll"
```

```
xdebug.remote_enable=on
```

```
xdebug.remote_handler=dbgp
```

```
xdebug.remote_host=localhost
```

```
xdebug.remote_port=9000
```

Перезапускаем Apache

В NetBeans IDE перейти в Tools-Options-PHP. На вкладке General в секции Command Line указать путь до php.exe. В секции Debugging прописать port 9000 и Session ID netbeans-xdebug.

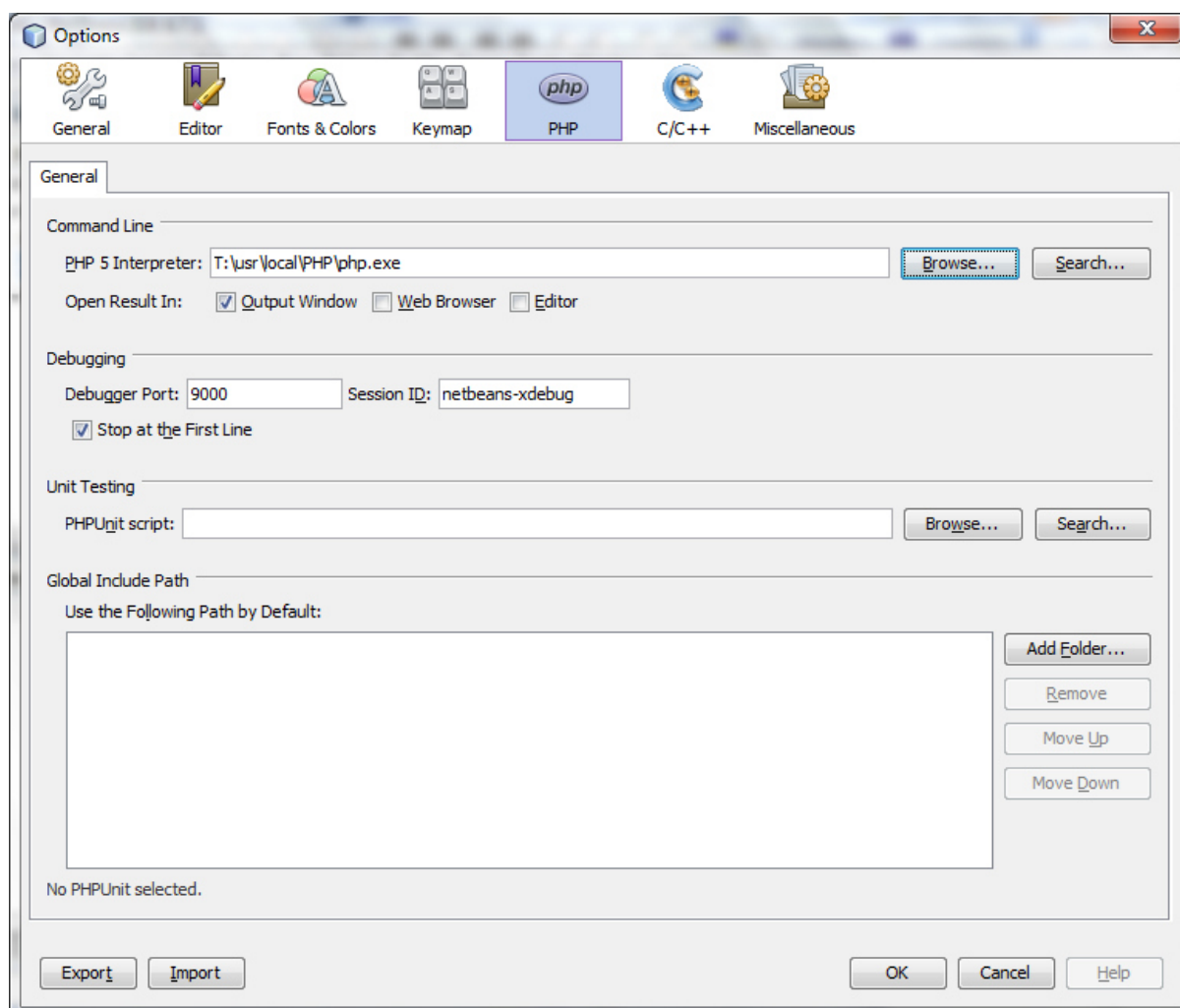


Рисунок 3.8 – Настройка отладчика

3.2.1 Как работает отладка

Когда отладка включена в php.ini, xdebug начинает контролировать выполнение программы, это значит, что xdebug может останавливать выполнение программы в любой момент, а потом запускать ее снова.

Когда выполнение программы остановлено, xdebug может получить информацию о текущем состоянии программы, например, прочитать текущие

значения переменных. Также xdebug представляет возможность изменять значения переменных во время выполнения скрипта.

Расширение xdebug – это сервер, ожидающие соединений с клиентами на определенном порту, задаваемом в конфигурации. Существует два протокола, которые могут быть использованы для коммуникации между xdebug-клиентом и xdebug-сервером (GDB и DBGp).

GDB – это старый протокол, который был заменен на DBGp. Посылая команды серверу-xdebug, xdebug-клиент удаленно контролирует выполнение, сообщает PHP о приостановке запуска, выполнении следующей строки кода или продолжении выполнения программы. Клиент обычно встроен в редактор или IDE (в нашем случае в NetBeans), поэтому вы не будете иметь дело с протоколом общения напрямую.

Веб-сервер с xdebug может быть запущен на другой операционной системе, нежели xdebug-клиент. Вот почему xdebug называется удаленным отладчиком(remote debugger). Для упрощения, мы настроим сервер и клиент на одном и том же компьютере.

Существует два режима начала сессии отладки. Эти режимы контролируются в php.ini настройкой xdebug.remote_mode. Значение по умолчанию req, xdebug-клиент будет соединяться с сервером всякий раз когда начинается выполнение скрипта. Если вы хотите, чтобы xdebug соединялся с сервером только тогда, когда установлен breakpoint или когда в скрипте возникла ошибка, то можно установить xdebug.remote_mode в jit.

Для того чтобы начать отладку, нужно послать параметр XDEBUG_SESSION_START в скрипт как часть GET, POST или COOKIE. Значение этого параметра это название сессии отладки, которое должно быть уникальным, по имени сессии xdebug может различать различные сессии, запущенные параллельно. Для завершения сессии отладки необходимо послать скрипту команду XDEBUG_SESSION_STOP.

Вместо того чтобы вручную начинать и заканчивать отладочные сессии, следует установить специальный firefox plugin, который поможет легко начинать и заканчивать сессии одним нажатием мышки.

					ПГУ 2.090301.36.001 ПЗ	Лист
						82
Изм.	Лист	№ докум	Подпись	Дата		

4. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ. ЭРГОНОМИКА РАБОЧЕГО МЕСТА ПРОГРАММИСТА.

4.1 Характеристика условий труда программиста

Технический прогресс глобально поменял условия производственной деятельности сотрудников умственного труда. Их труд стал более интенсивнее, требующим больших затрат умственной, эмоциональной и физической энергии. Это потребовало системного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

В настоящее время компьютеры широко применяется во всех областях деятельности человека. При работе с компьютером, человек подвергается воздействию ряда вредных электромагнитных полей (ВЧ, УВЧ и СВЧ), шума и вибрации, статического электричества и др.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной работы и большой нагрузкой на мышцы рук при работе с клавиатурой. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы программиста.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечаются болезненное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

Одним из критериев, по которому определяется «пригодность» рабочего места для постоянного пребывания сотрудников, является эргономические требования к рабочему месту и освещенность рабочего места программиста.

4.2 Эргономические требования к рабочему месту программиста

Проектирование рабочих мест, снабженных мониторами, относится к числу важных проблем эргономического проектирования в области вычислительной техники.

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия: оптимальное размещение оборудования, входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.

Эргономическими аспектами проектирования рабочих мест программиста, являются:

1. высота рабочей поверхности,
2. размеры пространства для ног,
3. требования к расположению документов на рабочем месте
(наличие и размеры подставки для документов, возможность различного размещения документов, расстояние от глаз пользователя до экрана, документа, клавиатуры и т.д.),
4. характеристики рабочего кресла,
5. требования к поверхности рабочего стола,
6. регулируемость элементов рабочего места.

Главными элементами рабочего места программиста являются стол и кресло.

Основным рабочим положением является положение сидя.

Рабочая поза сидя вызывает минимальное утомление программиста.

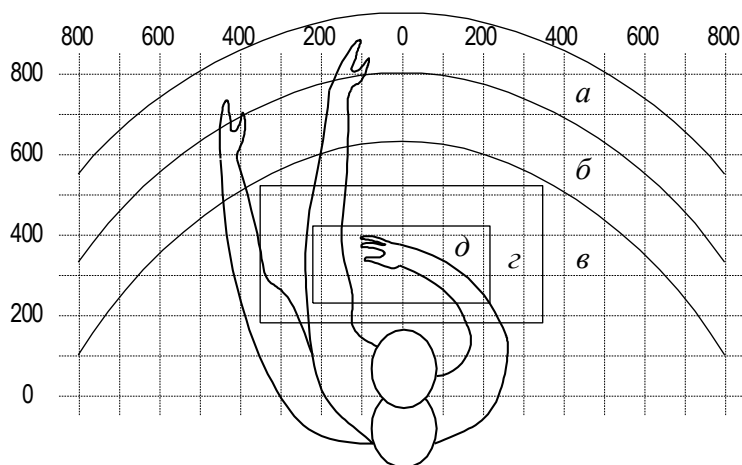
Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации.

То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Моторное поле - пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук - это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.



- а - зона максимальной досягаемости;
- б - зона досягаемости пальцев при вытянутой руке;
- в - зона легкой досягаемости ладони;
- г - оптимальное пространство для грубой ручной работы;

Рисунок 5.1 Зоны досягаемости рук в горизонтальной

Оптимальное размещение предметов труда и документации в зонах досягаемости:

Монитор размещается в зоне **а** (в центре);

Системный блок размещается в предусмотренной нише стола;

Клавиатура - в зоне **г/д**;

«**Мышь**» - в зоне **в** справа;

Сканер в зоне, **а/б** (слева);

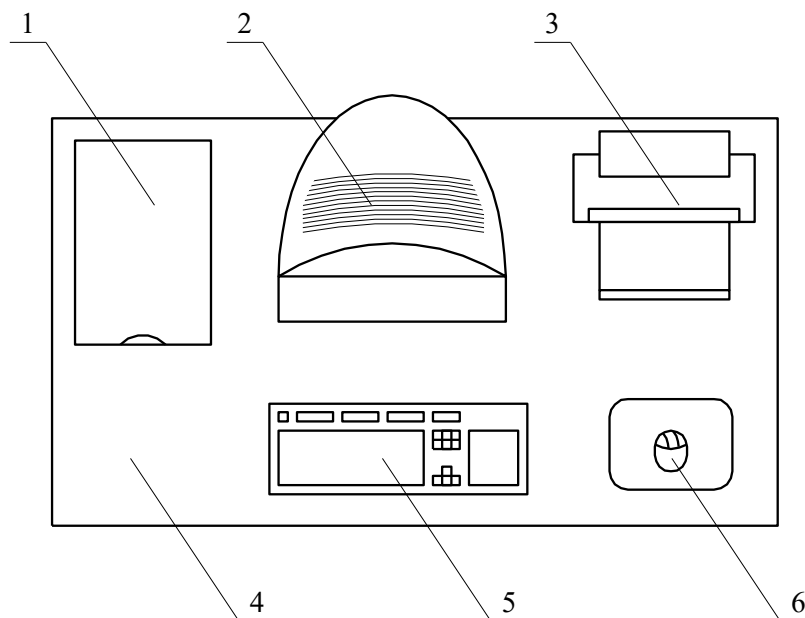
Принтер находится в зоне **а** (справа);

Документация необходимая при работе - в зоне легкой досягаемости ладони;

В выдвижных ящиках стола - литература, неиспользуемая постоянно.

На рис. 6.2 показан пример размещения основных и периферийных составляющих ПК на рабочем столе программиста.

1 – сканер, 2 – монитор, 3 – принтер, 4 – поверхность рабочего стола,



5 – клавиатура, 6 – «мышь».

Рисунок 6.2 Размещение основных и периферийных составляющих ПК.

Для комфортной работы стол должен удовлетворять следующим условиям:

1. высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;
2. нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;
3. поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;
4. конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, канцелярских принадлежностей).

5. высота рабочей поверхности рекомендуется в пределах 680-760 мм. Высота поверхности, на которую устанавливается клавиатура, должна быть около 650 мм.

Большое значение придается характеристикам рабочего кресла. Так, рекомендуемая высота сиденья над уровнем пола находится в пределах 420-550 мм. Поверхность сиденья мягкая, передний край закругленный, а угол наклона спинки - регулируемый.

Необходимо предусматривать при проектировании возможность различного размещения документов: сбоку от монитора, между монитором и клавиатурой и т.п. Кроме того, в случаях, когда монитор имеет низкое качество изображения, например заметны мелькания, расстояние от глаз до экрана делают больше (около 700 мм), чем расстояние от глаза до документа (300-450 мм). Вообще при высоком качестве изображения на видеотерминале расстояние от глаз пользователя до экрана, документа и клавиатуры может быть равным.

Положение экрана определяется:

1. расстоянием считывания (0,6...0,7 м);
2. углом считывания, направлением взгляда на 20° ниже горизонтали к центру экрана, причем экран перпендикулярен этому направлению.

Должна также предусматриваться возможность регулирования экрана:

3. по высоте +3 см;
4. по наклону от -10° до $+20^\circ$ относительно вертикали;
5. в левом и правом направлениях.

Большое значение также придается правильной рабочей позе пользователя.

При неудобной рабочей позе могут появиться боли в мышцах, суставах и сухожилиях. Требования к рабочей позе пользователя монитора следующие:

1. голова не должна быть наклонена более чем на 20° ,

2. плечи должны быть расслаблены,
3. локти - под углом $80^{\circ} \dots 100^{\circ}$,
4. предплечья и кисти рук - в горизонтальном положении.

Причина неправильной позы пользователей обусловлена следующими факторами: нет хорошей подставки для документов, клавиатура находится слишком высоко, а документы - низко, некуда положить руки и кисти, недостаточно пространство для ног.

В целях преодоления указанных недостатков даются общие рекомендации: лучше передвижная клавиатура; должны быть предусмотрены специальные приспособления для регулирования высоты стола, клавиатуры и монитора, а также подставка для рук.

Существенное значение для производительной и качественной работы на компьютере имеют размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана. Если расстояние от глаз оператора до экрана дисплея составляет $60 \dots 80$ см, то высота знака должна быть не менее 3 мм, оптимальное соотношение ширины и высоты знака составляет 3:4, а расстояние между знаками – $15 \dots 20\%$ их высоты.

Соотношение яркости фона экрана и символов - от 1:2 до 1:15.

Во время пользования компьютером медики советуют устанавливать монитор на расстоянии 50-60 см от глаз. Специалисты также считают, что верхняя часть видеодисплея должна быть на уровне глаз или чуть ниже. Когда человек смотрит прямо перед собой, его глаза открываются шире, чем когда он смотрит вниз. За счет этого площадь обзора значительно увеличивается, вызывая обезвоживание глаз. К тому же если экран установлен высоко, а глаза широко открыты, нарушается функция моргания. Это значит, что глаза не закрываются полностью, не омываются слезной

жидкостью, не получают достаточного увлажнения, что приводит к их быстрой утомляемости.

Создание благоприятных условий труда и правильное эстетическое оформление рабочих мест на производстве имеет большое значение, как для облегчения труда, так и для повышения его привлекательности, положительно влияющей на производительность труда.

4.3 Освещенность рабочего места программиста

Правильно спроектированное и выполненное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на работающего, повышает безопасность труда и снижает травматизм.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

Существует три вида освещения - естественное, искусственное и совмещенное (естественное и искусственное вместе).

Естественное освещение - освещение помещений дневным светом, проникающим через световые проемы в наружных ограждающих конструкциях помещений. Естественное освещение характеризуется тем, что меняется в широких пределах в зависимости от времени дня, времени года, характера области и ряда других факторов.

Искусственное освещение применяется при работе в темное время суток и днем, когда не удастся обеспечить нормированные значения коэффициента естественного освещения (пасмурная погода, короткий световой день). Освещение, при котором недостаточное по нормам естественное освещение дополняется искусственным, называется совмещенным освещением.

Искусственное освещение подразделяется на рабочее, аварийное, эвакуационное, охранное. Рабочее освещение, в свою очередь, может быть общим или комбинированным. Общее - освещение, при котором светильники размещаются в верхней зоне помещения равномерно или применительно к расположению оборудования. Комбинированное - освещение, при котором к общему добавляется местное освещение.

Согласно СНиП 23-05-2010 в помещениях вычислительных центров необходимо применить систему комбинированного освещения.

При выполнении работ требующих высокой зрительной точности (наименьший размер объекта различения 0,3...0,5 мм), величина коэффициента естественного освещения (КЕО) должна быть не ниже 1,5%, а при зрительной работе средней точности (наименьший размер объекта различения 0,5...1,0 мм) КЕО должен быть не ниже 1,0%.

В качестве источников искусственного освещения обычно используются люминесцентные лампы типа ЛБ или ДРЛ, которые попарно объединяются в светильники, которые должны располагаться над рабочими поверхностями равномерно.

Требования к освещенности в помещениях, где установлены компьютеры, следующие: при выполнении зрительных работ высокой точности общая освещенность должна составлять 300 лк, а комбинированная

- 750 лк; аналогичные требования при выполнении работ средней точности - 200 и 300 лк соответственно.

Кроме того, все поле зрения должно быть освещено достаточно равномерно – это основное гигиеническое требование. Иными словами, степень освещения помещения и яркость экрана компьютера должны быть примерно одинаковыми, т.к. яркий свет в районе периферийного зрения значительно увеличивает напряженность глаз и, как следствие, приводит к их быстрой утомляемости.

4.4 Расчет освещения рабочего места программиста

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

1. по спектральному составу света они близки к дневному, естественному свету;
2. обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
3. обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
4. более длительный срок службы.

Расчет освещения производится для комнаты площадью 15 м^2 , ширина которой 5 м, высота - 3 м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n}, \text{ где}$$

F - рассчитываемый световой поток, Лм;

E - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная

освещенность будет $E = 300$ Лк;

S - площадь освещаемого помещения (в нашем случае $S = 15 \text{ м}^2$);

Z - отношение средней освещенности к минимальной (обычно принимается равным $1,1 \dots 1,2$, пусть $Z = 1,1$);

K - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае $K = 1,5$);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризующих коэффициентами отражения от стен (P_C) и потолка (P_{Π}), значение коэффициентов P_C и P_{Π} были указаны выше: $P_C = 40\%$, $P_{\Pi} = 60\%$. Значение n определим по таблице коэффициентов использования

различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = \frac{S}{h \cdot (A + B)}, \text{ где}$$

S - площадь помещения, $S = 15 \text{ м}^2$;

h - расчетная высота подвеса, $h = 2.92 \text{ м}$;

A - ширина помещения, $A = 3 \text{ м}$;

B - длина помещения, $B = 5$ м.

Подставив значения получаем:

$$I = \frac{15}{2,92 \cdot (3+5)} = 0,64$$

Зная индекс помещения I , по таблице находим $n = 0,22$

Подставим все значения в формулу для определения светового потока F :

$$F = \frac{300 \cdot 1,5 \cdot 15 \cdot 1,1}{0,22} = 33750 \text{ Лм}$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых $F = 4320$ Лк.

Рассчитаем необходимое количество ламп по формуле:

$$N = \frac{F}{F_{\text{л}}}$$

N - определяемое число ламп;

F - световой поток, $F = 33750$ Лм;

$F_{\text{л}}$ - световой поток лампы, $F_{\text{л}} = 4320$ Лм.

$$N = \frac{33750}{4320} = 8 \text{ шт.}$$

При выборе осветительных приборов используем светильники типа ОД.
Каждый светильник комплектуется двумя лампами

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы было разработано web приложение, обеспечивающее реализацию онлайн магазина.

Данный проект это приложение на основе паттерна MVC, созданное на основе системы платежей PayPal. Приложение интегрируется со сторонними службами для обеспечения безопасности работы с данными. Приложение построено с использованием технологий, которые являются производительными, гибкими и масштабируемыми.

					ПГУ 2.090301.36.001 ПЗ	Лист
						95
Изм.	Лист	№ докум	Подпись	Дата		

ПРИЛОЖЕНИЕ А

ЛИСТИНГ

Файл index.php

```
include "db.php";
?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Simple Add to Cart</title>

    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <style type="text/css">

      .rwd-table {
margin: 1em 0;
min-width: 300px;
}
      .rwd-table tr {
border-top: 1px solid #ddd;
border-bottom: 1px solid #ddd;
}
      .rwd-table th {
display: none;
}
      .rwd-table td {
display: block;
}
      .rwd-table td:first-child {
padding-top: .5em;
}
      .rwd-table td:last-child {
padding-bottom: .5em;
}
      .rwd-table td:before {
content: attr(data-th) " : ";
color: "#000";
font-weight: bold;
width: 6.5em;
display: inline-block;
```

					ПГУ 2.090301.36.001 ПЗ	Лист
						96
Изм.	Лист	№ докум	Подпись	Дата		

```

    }
    @media (min-width: 513px) {
      .rwd-table td:before {
        display: none;
      }
    }
    .rwd-table th, .rwd-table td {
      text-align: left;
    }
    @media (min-width: 513px) {
      .rwd-table th, .rwd-table td {
        display: table-cell;
        padding: .25em .5em;
      }
      .rwd-table th:first-child, .rwd-table td:first-child {
        padding-left: 0;
      }
      .rwd-table th:last-child, .rwd-table td:last-child {
        padding-right: 0;
      }
    }
    .rwd-table {
      border-radius: .4em;
      overflow: hidden;
    }
    .rwd-table tr {
      border-color: #46637f;
    }
    .rwd-table th, .rwd-table td {
      margin: .5em 1em;
    }
    @media (min-width: 513px) {
      .rwd-table th, .rwd-table td {
        padding: 1em !important;
      }
    }
  }
</style>
</head>
<body>
  <div class="container">
    <div class="row">
      <div class="col-md-10 col-md-offset-1">

        <table class='table rwd-table'>

          <tr>

```

```

        <th > Product Name </th>
        <th > Product Details </th>
        <th > Product Price </th>
        <th > Product Quantity </th>
        <th > Product Date </th>
        <th > Action </th>
    </tr>

    <?php
    $result = $conn->query('select * from products');
    while($row = $result->fetch_object()){ ?>
    <tr>

        <td data-th="Product Name"> <?php echo $row-
>name; ?></td>

        <td data-th="Product Details"> <?php echo
$row->description; ?></td>

        <td data-th="Product Price"> <?php echo
$row->price; ?></td>

        <td data-th="Product Quantity"> <?php echo
$row->quantity; ?></td>

        <td data-th="Product Date"> <?php echo $row-
>date_created; ?></td>

        <td data-th="Action"><a class="btn btn-
primary" href="add_to_cart.php?id=<?php echo $row->id; ?>" >Add to Cart</a></td>
    </tr>
    <?php } ?>

    </table>
</div>
</div>
</div>
</body>
</html>

```

Файл Cart.php

```

class Cart
{
    protected static $requiredParams = array(
        'id',
        'name',
        'quantity',
        'price',
    );
    protected $id;

```

```

protected $identifier;
protected $store;
protected $currency;
public function __construct(StorageInterface $store, IdentifierInterface $identifier)
{
    $this->store = $store;
    $this->identifier = $identifier;
    $this->id = $this->identifier->get();
    if (method_exists($this->store, 'restore')) {
        $this->store->restore($this->id);
    }

    $this->store->setIdentifier($this->id);
}
public function __destruct()
{
    if (method_exists($this->store, 'save')) {
        $this->store->save();
    }
}
protected function checkArgs(array $item)
{
    foreach (static::$requiredParams as $param) {
        if (!array_key_exists($param, $item)) {
            throw new \InvalidArgumentException("The '{$param}' field is required");
        }
    }
}
public function &contents($asArray = false)
{
    return $this->store->data($asArray);
}
public function &contentsArray()
{
    return $this->store->data(true);
}
protected function createItemIdentifier(array $item)
{
    if (!array_key_exists('options', $item)) {
        $item['options'] = array();
    }
    ksort($item['options']);
    return md5($item['id'] . serialize($item['options']));
}
public function currency()
{

```

```

        return $this->currency;
    }
    */
    public function destroy()
    {
        $this->store->destroy();
    }
    {
        return $this->store->find($id);
    }
    public function getIdentifier()
    {
        return $this->id;
    }
    public function has($itemIdentifier)
    {
        return $this->store->has($itemIdentifier);
    }

    public function insert(array $item)
    {
        $this->checkArgs($item);
        $itemIdentifier = $this->createItemIdentifier($item);
        if ($this->has($itemIdentifier)) {
            $item['quantity'] = $this->item($itemIdentifier)->quantity + $item['quantity'];
            $this->update($itemIdentifier, $item);
            return $itemIdentifier;
        }
        if ($item['quantity'] < 1) {
            throw new \InvalidArgumentException('Quantity can not be less than 1');
        }
        $item = $this->createItem($itemIdentifier, $item);
        $this->store->insertUpdate($item);
        return $itemIdentifier;
    }
    protected function createItem($identifier, array $data)
    {
        return new Item($identifier, $data, $this->store);
    }
    public function item($itemIdentifier)
    {
        return $this->store->item($itemIdentifier);
    }
    public function remove($identifier)
    {
        $this->store->remove($identifier);
    }

```

```

    }
    public function setCurrency($currency)
    {
        $this->currency = $currency;
        return $this;
    }
    public function setIdentifier($identifier)
    {
        $this->store->setIdentifier($identifier);
    }
    public function tax()
    {
        $total = 0;
        foreach ($this->contents() as $item) {
            $total += (float)$item->tax();
        }
        return $total;
    }
    public function total($includeTax = true)
    {
        $total = 0;
        foreach ($this->contents() as $item) {
            $total += (float)$item->total($includeTax);
        }
        return (float)$total;
    }
    public function totalWithTax()
    {
        return $this->total(true);
    }
    public function totalWithoutTax()
    {
        return $this->total(false);
    }
    public function totalItems($unique = false)
    {
        $total = 0;
        foreach ($this->contents() as $item) {
            $total += $unique ? 1 : $item->quantity;
        }
        return $total;
    }
    public function totalUniqueItems()
    {
        return $this->totalItems(true);
    }
}

```

```

public function update($itemIdentifier, $key, $value = null)
{
    foreach ($this->contents() as $item) {
        if ($item->getIdentifier() == $itemIdentifier) {
            $item->update($key, $value);
            break;
        }
    }
}

```

Файл Item.php

```

class Item
{
    protected $identifier;
    protected $store;
    protected $tax;
    protected $data = array();
    public function __construct($identifier, array $item, StorageInterface $store)
    {
        $this->identifier = $identifier;
        $this->store = $store;
        foreach ($item as $key => $value) {
            $this->data[$key] = $value;
        }
        $item['tax'] = isset($item['tax']) ? $item['tax'] : 0;
        $this->tax = new Tax($item['tax']);
    }
    public function __get($param)
    {
        if ($param == 'identifier') {
            return $this->identifier;
        }
        if (isset($this->$param) === true) {
            return $this->data[$param];
        } else {
            return null;
        }
    }
    public function __isset($param)
    {
        if ($param == 'identifier') {
            return true;
        }
    }
}

```

```

    }
    return array_key_exists($param, $this->data);
}
public function __set($param, $value)
{
    $this->data[$param] = $value;
}
public function getIdentifier()
{
    return $this->identifier;
}

public function hasOptions()
{
    return array_key_exists('options', $this->data) && !empty($this->data['options']);
}
public function remove()
{
    $this->store->remove($this->identifier);
}
public function tax($single = false)
{
    $quantity = $single ? 1 : $this->quantity;
    return $this->tax->rate($this->price * $quantity);
}
public function toArray()
{
    return $this->data;
}
public function total($includeTax = true)
{
    $price = $this->price;
    if ($includeTax) {
        $price = $this->tax->add($price);
    }
    return (float)($price * $this->quantity);
}
public function update($key, $value = null)
{
    if (is_array($key)) {
        foreach ($key as $updateKey => $updateValue) {
            $this->update($updateKey, $updateValue);
        }
    } else {
        $key = (string)$key;
        if ($key == 'quantity') {

```

```

        //
        if ($value <= 0) {
            $this->remove();
            return;
        }
    }
    if ($key == 'tax' && is_numeric($value)) {
        $this->tax = new Tax($value);
    }
    $this->data[$key] = $value;
}
}
}

```

Файл Login.php

```

<?php
<?php require_once 'app/init.php';require 'app/parts/header.php'; ?>
$validate = new Validate();
$user = new User();
if(Input::exists()){
    $login = $user->login('username', 'password');
    try{
        if($login){
            if(Cookie::exists('cookie_name')){
                $cart = new Cart();
                $cart->update();
                Cookie::delete('cookie_name');
            }
            header('Location: http://localhost/ShoppingCart/');
        }
    }catch(Exception $e){
        $e->getMessage();
    }
}
?>
<div class="login_form">
    <form method="post" class="form-horizontal">
        <div class="form-group">
            <label class="control-label col-sm-2" for="username">Username:</label>
            <div class="col-sm-10">
                <input type="text" class="form-control" id="username" name="username"
placeholder="Enter username" value="<?php echo Input::get('username'); ?>"><span><?php echo
$user->error(0); ?></span>
            </div>

```

```

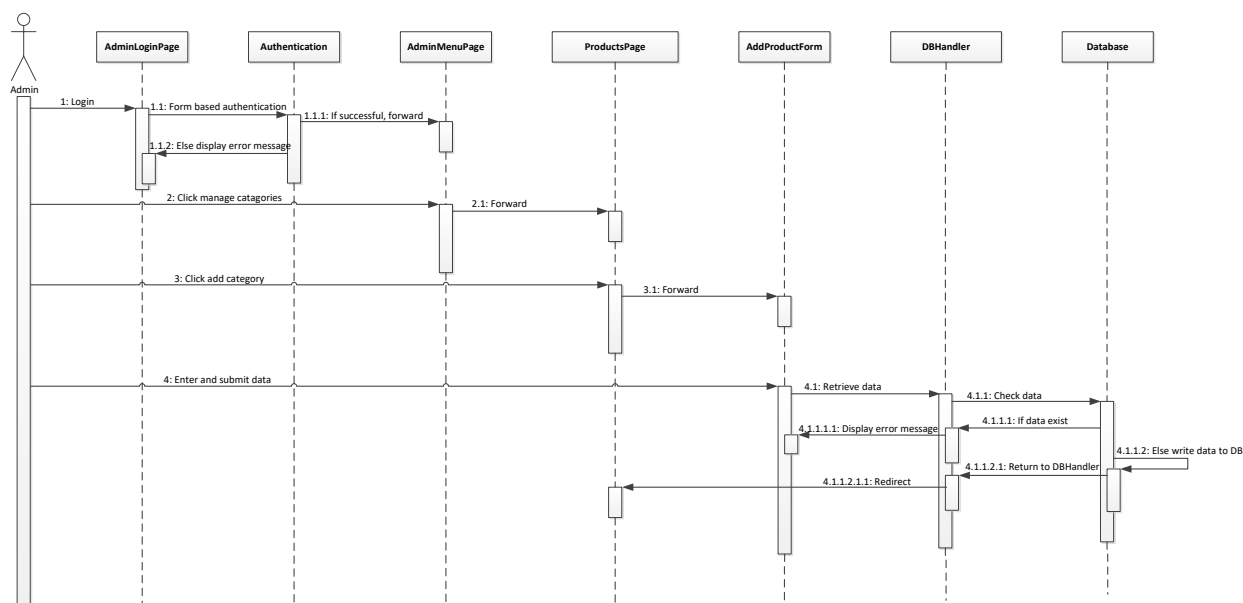
        </div>
        <div class="form-group">
            <label class="control-label col-sm-2" for="password">Password:</label>
            <div class="col-sm-10">
                <input type="password" class="form-control" id="password" name="password"
placeholder="Enter password" autocomplete="off" <?php echo $user->error(0); ?><span><?php
echo $user->error(1); ?></span>
            </div>
        </div>

        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <button type="submit" class="btn btn-default" name="login">Login</button>
            </div>
        </div>
    </form>
</div>
<?php require 'app/parts/footer.php'; ?>

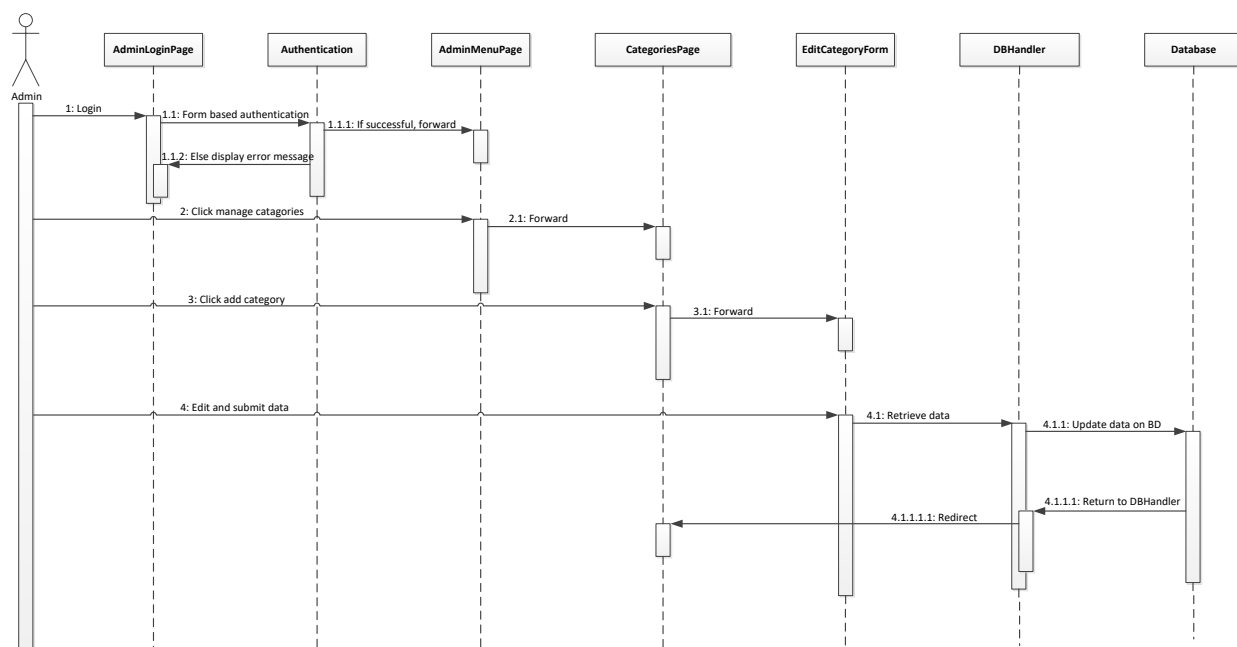
```

ПРИЛОЖЕНИЕ Б

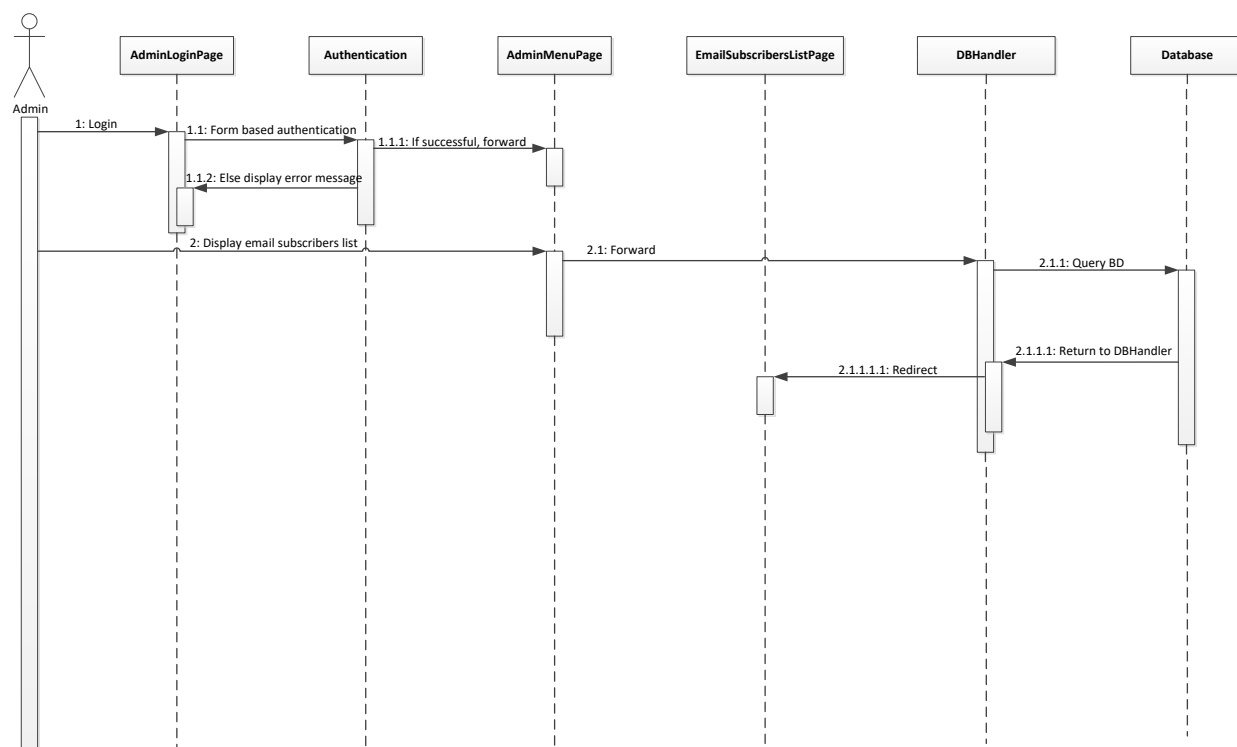
UML ОПИСАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ В СИСТЕМЕ ОНЛАЙН ТОРГОВЛИ



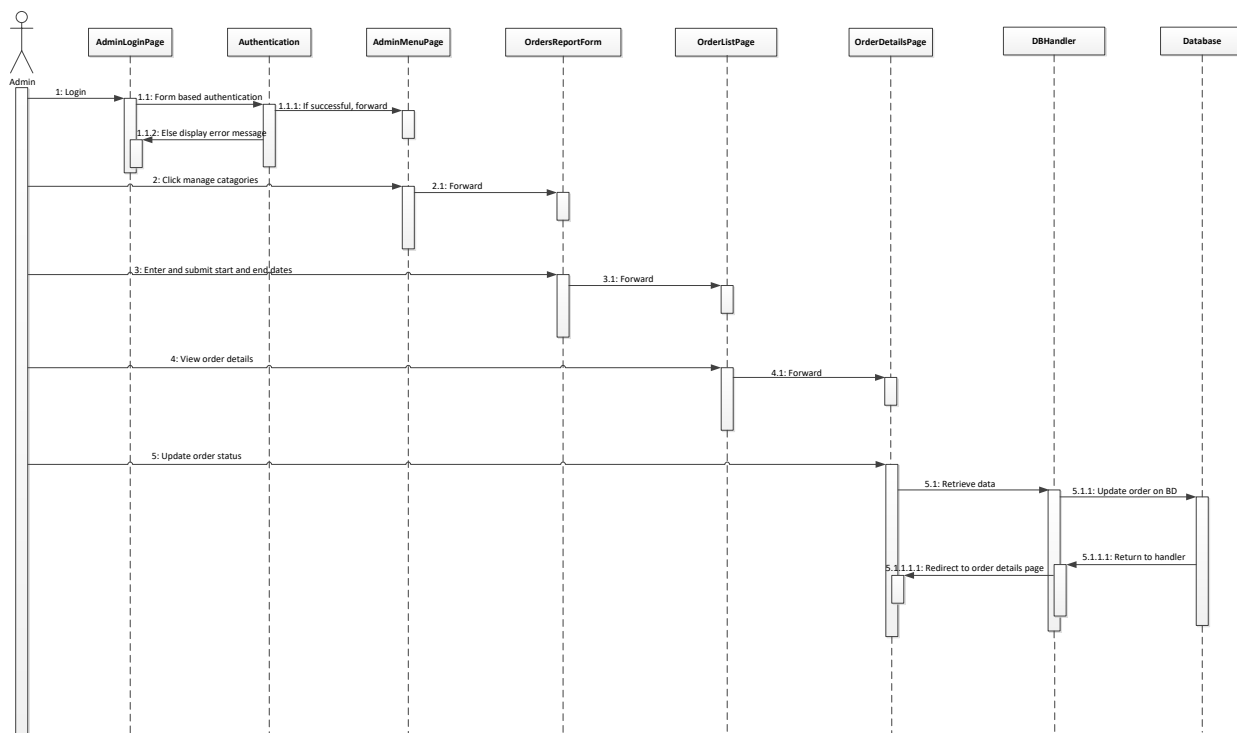
Получение данных о каталогах из СУБД



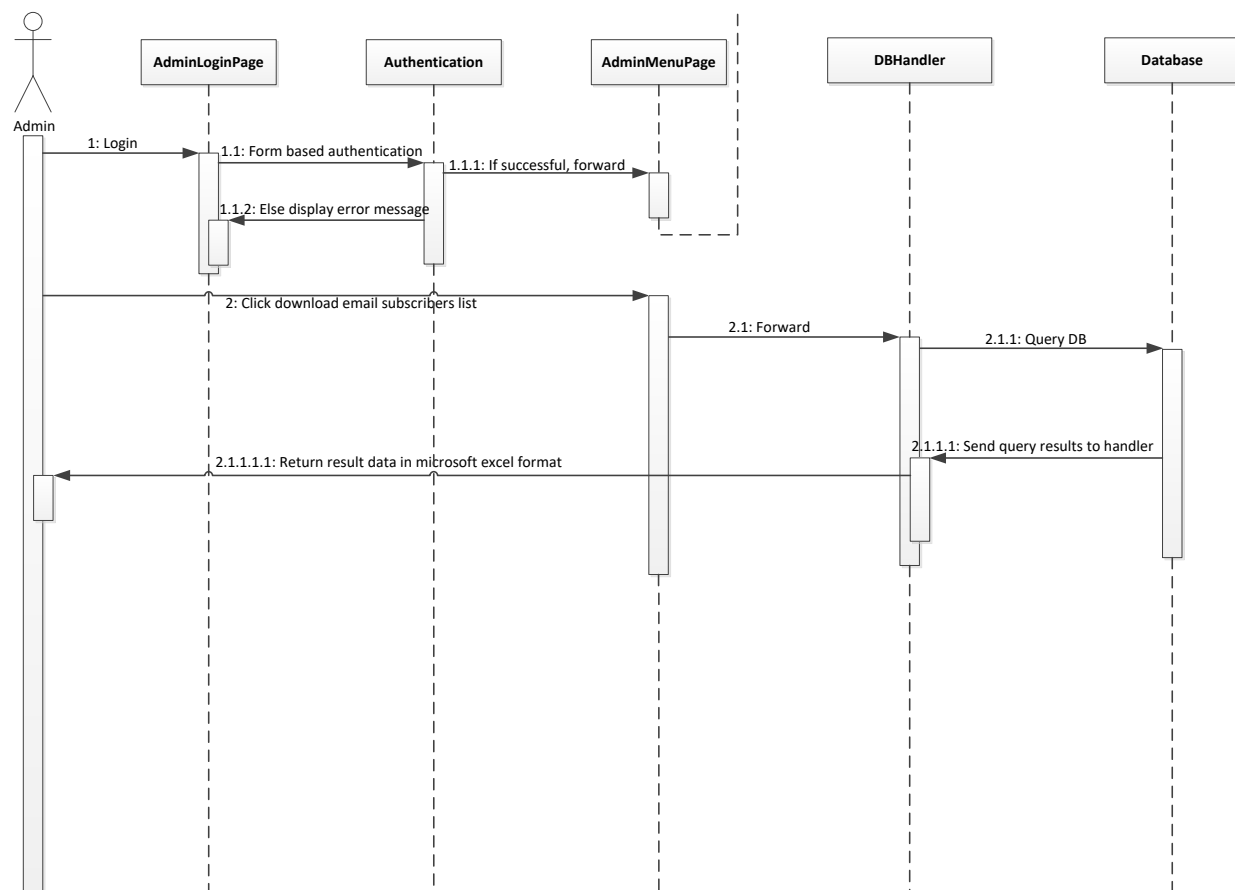
Работа с каталогами



Получение списка товара



Обновление статуса заказа



Вход в систему

ПРИЛОЖЕНИЕ В

Презентация

Тема работы Система онлайн торговли «бизнес-потребитель»

АВТОР РАБОТЫ: СТУДЕНТ ГРУППЫ 16ВВ1 ТИМИРЯЩЕВ А.В

НАУЧНЫЙ РУКОВОДИТЕЛЬ Д.Т.Н., ПРОФ.КАФ ВТ ЗИНКИН С.А



Введение

В современном мире электронная торговля занимает одно из ведущих мест ведения бизнеса: она обеспечивает необходимыми товарами и услугами, объединяет континенты и страны.

Большое число людей, особенно молодежь, очень часто предпочитает покупку товаров в Интернете, нежели в традиционных магазинах. Прежде всего, это объясняется приемлемыми ценами, а также более широким ассортиментом товара. Помимо этого затрата времени на совершение покупки посредством интернет-технологий гораздо меньше.

Несмотря на ряд преимуществ совершения онлайн покупок, существуют моменты онлайн системы требующие проработки: разработка интуитивно понятного интерфейса, разработка защищенной системы оплаты, разработка поисковой и рекомендательных систем внутри системы, разработки системы поиска контентной информации в интернете



Функционал приложения

В процессе выполнения работы планируется проработать следующие вопросы:

- Разработать СУБД для хранения каталога товаров, категории продукты и иную информацию.
- Разработать структурированные запросы (SQL) и программный код (в рамках выбранного инструментария) для доступа к данным и сделать бизнес процессы ресурса функциональными.
- Добавить данные в каталог продуктов, которые определяют атрибуты продукта.
- Разработать систему платежей в онлайн системе платежей (например, PayPal).
- Разработать для администраторов сайта личный раздел сайта, где они могут управлять каталогами товаров.
- Реализовать функцию «Корзина покупателя».
- Создать страницу администрирования заказа.
- Внедрить систему рекомендаций по продукту.



Постановка задачи

Цель работы – разработать масштабируемую, отказоустойчивую систему продажи товаров на базе паттерна модель, вид, контроллер.

Объектом разработки является программное приложение, обеспечивающее клиентам доступ к каталогу товаров и услуг, администраторам доступ к настройкам системы на базе трехзвенной архитектуры.



					ПГУ 2.090301.36.001 ПЗ	Лист
						110
Изм.	Лист	№ докум	Подпись	Дата		

Шаблон MVC описывает простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса. В результате, приложение легче масштабируется, тестируется, сопровождается и конечно же реализуется.



The diagram illustrates the functional requirements of an online store system, organized into three swimlanes: Administrator, Customer, and Payment System.

Administrator Swimlane:

- Use Cases:
 - Добавить каталог и продукт (Add catalog and product)
 - Редактировать каталог и продукт (Edit catalog and product)
 - Удалить каталог и продукт (Delete catalog and product)
 - Просмотр всех категорий и товаров в списках (View all categories and goods in lists)
 - Просмотр списка платежей (View payment list)
 - Генерировать счет за платеж (Generate bill for payment)
 - Просмотр списка e-mail подписчиков (View subscriber e-mail list)
 - Выгрузка e-mail подписчиков (Export subscriber e-mail)
 - Просмотр выбранного платежа (View selected payment)
- Relationships:
 - «include» relationships from the first six use cases to the central use case: **Вход в систему** (System login).
 - «extend» relationships from the last three use cases to **Обновить статус заказа** (Update order status).

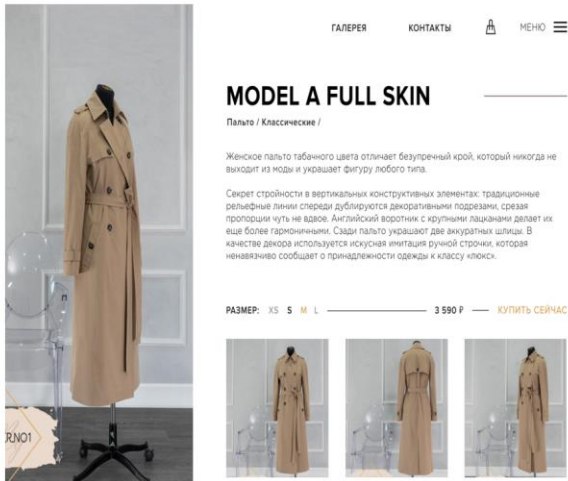
Customer Swimlane:

- Use Cases:
 - Просмотр списка доступных товаров (View list of available goods)
 - Добавить продукт в корзину (Add product to cart)
 - Обновить корзину (Update cart)
 - Удалить товар из корзины (Remove item from cart)
 - Очистить корзину (Clear cart)
 - Оформить заказ (Place order)
 - Оформить заказ из корзины с помощью PayPal Express Checkout (Place order from cart using PayPal Express Checkout)
- Relationships:
 - «extend» relationships from the last two use cases to **Получить письмо с подтверждением заказа** (Receive order confirmation email).
 - «extend» relationship from the last use case to **Подтверждение оплаты по e-mail** (Payment confirmation by email).

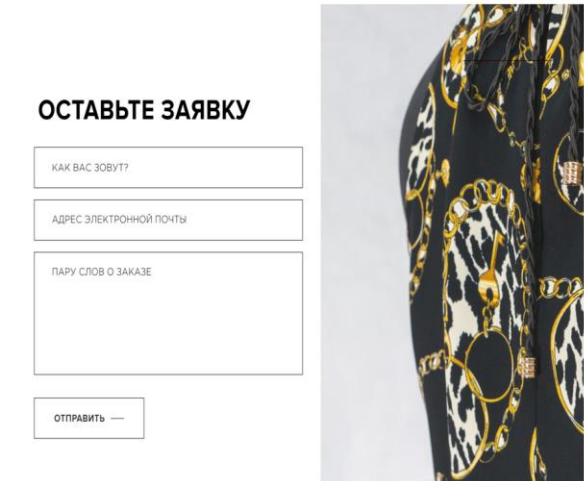
Payment System Swimlane:

- Use Case:
 - Платежная система (Payment system)
- Relationships:
 - «extend» relationship from the use case to **Подтверждение оплаты по e-mail**.

Пользовательский интерфейс



Выбор товара



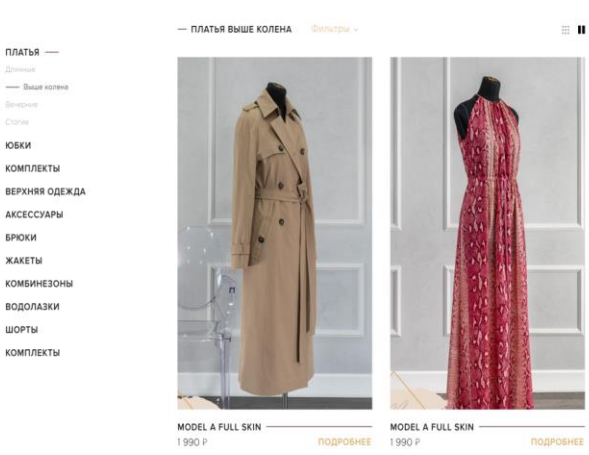
Оставить заявку



Пользовательский интерфейс



Главное окно



Основная галерея



Изм.	Лист	№ докум	Подпись	Дата

ПГУ 2.090301.36.001 ПЗ

Пользовательский интерфейс

ГАЛЕРЕЯ

КОНТАКТЫ

МЕНЮ

≡

ГАЛЕРЕЯ

КОНТАКТЫ

МЕНЮ

≡

ОФОРМЛЕНИЕ ЗАКАЗА

PayPal

1 000 Р

Ship to

John Doe

Russia, Moscow, Tverskaya 18

☐ Make this my preferred shipping address

Change >

Pay with

PayPal Balance

preferred

MasterCard x-2114

+

Add a debit or credit card

Manage >

PayPal is the safer, faster way to pay

No matter where you shop, we keep your financial information secure.

Pay Now

Cancel and return to Gearbest

Legal User Agreement Privacy Feedback © 1999 - 2019

ОФОРМЛЕНИЕ ЗАКАЗА

Ваш заказ №311230007 успешно оформлен и оплачен, вы получите уведомление, когда можно будет его забрать!

ПРОДОЛЖИТЬ ПОКУПКИ

Заказ оформлен

Платеж PayPal



Пользовательский интерфейс

ГАЛЕРЕЯ

КОНТАКТЫ

МЕНЮ

≡

ГАЛЕРЕЯ

КОНТАКТЫ

МЕНЮ

≡

КОРЗИНА

ПРОДУКТ

ЦВЕТ

РАЗМЕР

КОЛИЧЕСТВО

ЦЕНА

Model A full skin

Красный

XS

~ 1 +

1 000 Р

УДАЛИТЬ

СУММА К ОПЛАТЕ: 1 000 Р

ОФОРМИТЬ ЗАКАЗ

ОФОРМЛЕНИЕ ЗАКАЗА

Андрей

tim@gmail.com

КОММЕНТАРИЙ К ЗАКАЗУ

СПОСОБ ДОСТАВКИ

СУММА К ОПЛАТЕ: 1 700 Р

ОПЛАТИТЬ

Выберите способ доставки чтобы оплатить

Корзина

Оформление заказа



Заключение

В ходе выполнения ВКР было разработано web приложение обеспечивающее реализацию онлайн магазина. Приложение интегрируется со сторонними службами для обеспечения безопасности работы с данными. Данный проект это приложение на основе паттерна MVC, построен с использованием технологий php, mysql, apache, которые являются производительными, гибкими и масштабируемыми.



Спасибо за внимание!



					ПГУ 2.090301.36.001 ПЗ	Лист
						114
Изм.	Лист	№ докум	Подпись	Дата		