**АННОТАЦИЯ**

Отчёт о курсовой работе: 66 с., 20 рис., 1 табл., 2 приложения, 9 источников.

Объект исследования – детские игрушки.

Предмет исследования – функционирование и деятельности сайта магазина по продажам детских игрушек.

Цель работы – разработать и реализовать сайт магазин по продаже детских игрушек.

Метод исследования – анализ возможностей фреймворка Bootstrap, языка программирования JavaScript, языка PHP для создания и обработки баз данных и разработки серверных приложений.

В курсовой работе был разработан сайт магазин по продаже детских игрушек.

Сайт позволит просматривать информацию о различных видах игрушек, добавлять в корзину, оставлять заказы на товары.

Дальнейшее развитие системы связано с расширением возможностей сайта, изменение работы с заказами, вывода различных результатов на основании заказов и сообщений клиентов.

МАГАЗИН, ДЕТСКАЯ ИГРУШКА, САЙТ, OPEN SERVER, APACHE, СЕРВЕР БАЗ ДАННЫХ MYSQL, PHPMYADMIN, PHP, BOOTSTRAP, HTML, JAVASCRIPT, CSS.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc39165952)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc39165953)

[1.1 Состояние вопроса 7](#_Toc39165954)

[1.2 Моделирование существующих бизнес-процессов 7](#_Toc39165955)

[1.3 Актуальность и цель работы 9](#_Toc39165956)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 11](#_Toc39165957)

[2.1 Описание области применения и исходных данных приложения 11](#_Toc39165958)

[2.2 Требования к пользовательским интерфейсам 11](#_Toc39165959)

[2.3 Требования к проектированию программных интерфейсов 11](#_Toc39165960)

[2.4 Требования к аппаратным и программным интерфейсам 13](#_Toc39165961)

[2.5 Требования к пользователям продукта 14](#_Toc39165962)

[2.6 Функции продукта 14](#_Toc39165963)

[2.7 Ограничения 14](#_Toc39165964)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ 15](#_Toc39165965)

[3.1 Язык программирования серверных скриптов PHP 15](#_Toc39165966)

[3.2 Сервер баз данных MYSQL 17](#_Toc39165967)

[3.3 WEB-сервер APACHE 18](#_Toc39165968)

[3.4 Набор инструментов Bootstrap 19](#_Toc39165969)

[3.5 Язык программирования JAVASCRIPT 19](#_Toc39165970)

[3.6 Серверная и программная платформа Open Server 22](#_Toc39165971)

[3.7 Архитектура «клиент-сервер» 22](#_Toc39165972)

[4 РАЗРАБОТКА БАЗЫ ДАННЫХ САЙТА МАГАЗИНА ПО ПРОДАЖЕ ДЕТСКИХ ИГРУШЕК 24](#_Toc39165973)

[4.1 Выбор и обоснование CУБД 24](#_Toc39165974)

[4.2 Концептуальное проектирование БД 26](#_Toc39165975)

[4.3 Физическое проектирование БД 27](#_Toc39165976)

[5 САЙТ МАГАЗИН ПО ПРОДАЖЕ ДЕТСКИХ ИГРУШЕК 29](#_Toc39165977)

[5.1 Входные и выходные данные 29](#_Toc39165978)

[5.2 Структура сайта 29](#_Toc39165979)

[5.3 Верстка сайта 30](#_Toc39165980)

[5.4 Описание алгоритма работы скриптов и SQL-запросов сайта 31](#_Toc39165981)

[6 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 36](#_Toc39165982)

[6.1 Аппаратные, системные и программные требования 36](#_Toc39165983)

[6.2 Описание установки и запуска сайта 36](#_Toc39165984)

[6.3 Установка, настройка и запуск программы для работы с базами данных 38](#_Toc39165985)

[ЗАКЛЮЧЕНИЕ 41](#_Toc39165986)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 42](#_Toc39165987)

[ПРИЛОЖЕНИЕ А 43](#_Toc39165988)

[ПРИЛОЖЕНИЕ Б 44](#_Toc39165989)

# ВВЕДЕНИЕ

Почти каждая компании современности считает необходимым иметь страничку в интернете, и чем более упрощающей работу сотрудникам и клиентам она будет, тем лучше. Интернет-магазины имеют много преимуществ перед стационарными торговыми точками. И если раньше они использовались преимущественно, как подспорье имеющимся реальным магазинам. То сейчас многие люди, начиная бизнес, создают только интернет-магазины и зарабатывают с помощью них свои капиталы [7].

Одним из главных преимуществ интернет-магазина является то, что он, будучи полноценным сайтом, содержит полную информацию о компании. Это является дополнительным методом рекламы для вашего бизнеса. Притом к этим данным, содержащимся на сайте, у пользователя есть доступ круглосуточно. Чего не скажешь о рекламных листовках или визитках, которые всегда теряются или выбрасываются [7].

Важно, в первую очередь, создать такой интернет-магазин, который будет привлекателен для пользователей, будет вызывать у них положительные эмоции и доверие [7].

В данной курсовой работе будет реализована возможность приёма заказов на приобретение детских игрушек с записью их в базу данных.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Состояние вопроса

Покупая игрушку в интернет-магазине, каждый современный родитель ставит перед собой здравомыслящие вопросы, прежде всего, зачем эта игрушка моему ребенку? Чему она его научит? Какую модель поведения будет провоцировать? Заинтересует ли она ребенка надолго или оставит равнодушным? Как выбрать качественную игрушку? Какие игрушки лучше купить годовалому малышу?

Во все времена мягкие игрушки были любимыми у детей и взрослых. Да, да, взрослые тоже любят получать в подарок тигрят, котиков и зайчиков независимо от возраста. Ведь любой плюшевый мишка, полученный на день рождения или другой праздник, – это ощущение заботы и внимания от близких или друзей.

Что тогда говорить о малышах? Ведь ребенок в образе игрушки приобретает себе помощника и друга, с которым весело, когда остаёшься один в комнате, или не страшно, если проводишь весь день в саду без мамы. А еще с ним приятно засыпать в обнимку вместо подушки, прикоснувшись щёчкой к пушистому покрытию.

Выбирая игрушку, необходимо определиться, для каких целей она необходима? Если усадить на диван для интерьера для взрослого – покупайте побольше, для сада – лучше взять небольшую, которую можно схватить за ручку или лапку. Хотим отметить, что детские игрушки должны быть безопасными, т.е. иметь сертификаты качества. Если они безвредны, то и игра с ними приносит сплошное удовольствие и никакого разочарования.

Мягкая игрушка формирует тактильные восприятия, учит распознавать на ощупь материалы, фактуры. В любом случае такой подарок принесет малышу и взрослому море позитивных эмоций, поднимет настроение. Отдельную категорию составляют мягкие рюкзаки в виде животных. Он непременно понравится любому чаду, ведь игры с ним, в него помещаются мелкие детские игрушки и другие мелочи [8].

На данный момент существует множество интернет-магазинов детских игрушек, ниже представлены примеры некоторых из них.

Одними из самых известных магазинов игрушек в украинском сегменте Интернета являются:

[**BabyHappy.com.ua - магазин классных детских игрушек**](https://babyhappy.com.ua/)

Рисунок 1.1 – BabyHappy.com.ua

BabyHappy.com.ua - крупный интернет-магазин. Специализируется на товарах для детей разных возрастов - от младенцев до школьников. Обладает широким ассортиментом. Занимается реализацией детской продукции известных брендов.

[**Panama™ - самый детский интернет-магазин в Киеве и всей Украине**](https://panama.ua/)

Рисунок 1.2 – Panama™

PANAMA.UA - один из самых объемных интернет-маркетов детской продукции. Реализует товары для деток, начиная от новорожденных и до школьников. Ассортимент достаточно широкий, присутствует продукция крупных известных производителей [2].

## Моделирование существующих бизнес-процессов

Проведём описание бизнес-процессов с помощью методологии IDEF0. Этот этап является важным по той причине, что моделирование бизнес-процессов позволяет оценить работу системы полностью. За счёт моделирования можно проследить, что происходит в процессах от начала, до завершения. Моделирование позволяет получить «внешний» взгляд на процессы и определить улучшения, которые повысят их эффективность. IDEF0 позволяет создать модель функций процесса [9]. Для описания бизнес-процессов для данного курсового проекта была выбрана нотация IDEF0, которая позволяет описать процессы в их функциональном аспекте. На диаграмме IDEF0 отображаются основные функции процесса, входы, выходы, управляющие воздействия и устройства, взаимосвязанные с основными функциями.

IDEF0 используется для создания функциональной модели, отображаю- щей структуру и функции системы, а также потоки информации и матери- альных объектов, связывающие эти функции.

Модель состоит из двух элементов – это дуги и блоги, с помощью первых описываются процессы. Методология подчиняется правилам сторон:

1) стрелка входа приходит всегда в левую кромку активности;

2) стрелка управления – в верхнюю кромку;

3) стрелка механизма – нижняя кромка;

4) стрелка выхода – правая кромка.

На рисунке 1.4 изображена контекстная диаграмма. Согласно правилам, она состоит из следующих элементов:

1) собственно процесс – запись данных в базу данных;

2) стрелка входных данных – это данные с сайта;

3) стрелка механизма – это обработка информации серверной стороной и сайтом;

4) стрелка выхода – записанные данные в базу данных.

Рисунок 1.3 – Контекстная диаграмма

Диаграмма декомпозиции блока А0 приведена на рисунке 1.5. Она состоит из следующих элементов:

1) блоки сайта, сервера и базы данных;

2) стрелка входных данных – это данные с сайта;

3) стрелка механизма – это JavaScript, PHP и MySQL;

4) стрелка выхода – записанные данные в базу данных.

Рисунок 1.4 – Диаграмма декомпозиции блока А0

## Актуальность и цель работы

Все родители знакомы с ситуацией, когда ребенок часто просит новую игрушку. Ведь интерес быстро пропадает, а желание иметь все и сразу – у детей остается. Поэтому родители, чтобы удовлетворить запросы малышей, ищут возможность выгодно и недорого купить детские игрушки. Кто-то по-прежнему остается верным магазинам и супермаркетам. Но, стоит учитывать, что стоимость игрушек, и других товаров, там всегда выше. Это объясняется тем, сто владельцы магазинов вынуждены кроме налогов платить зарплату сотрудникам, уборщикам помещений, оплачивать коммунальные услуги, и многое другое. Все эти расходы в будущем ложатся на плечи покупателей, ведь их закладывают в конечную стоимость товара. Другой вариант – интернет магазин. Здесь нет больших расходов и траты средств, а значит, стоимость товара будет значительно ниже. Многие потребители давно выбрали для себя такой способ покупок [7].

Анализ сайтов продажи детского игрового товара показал, что в подобных сервисах не утихает потребность, не смотря на обширную конкуренцию и, хотя, офлайн магазины всё ещё являются более предпочтительным видом покупки данного вида товара, интернет-магазины не остаются в стороне, так как многие у многих может не оказаться достаточно времени для заказа необходимой по своим внешним характеристикам детской игрушки.

Цель работы: создание сайта, позволяющего клиентам выбрать из спектра товаров и заказать определённую детскую игрушку, добавить сообщение.

Задачи, которые необходимо решить для реализации проекта:

1) изучить источники относительно отрасли детских игрушек в целом;

2) изучить подходы к реализации сайтов по продаже детских игрушек;

3) оценить существующие готовые сайты и их функционал;

4) составить предполагаемый функционал программного комплекса;

5) сформировать техническое задание(ТЗ), на основе которого будет разработан программный комплекс;

6) выбрать и изучить технологии для реализации поставленных в ТЗ задач;

7) проанализировать существующие аналоги систем общения с клиентами;

8) спроектировать базу данных (БД) на основе технического задания;

9) выбрать подходящую систему для управления базой данных;

10) спроектировать дизайн веб-сайта магазина, протестировать адаптивность страниц;

11) разработать программный комплекс;

12) провести тестирование программного комплекса;

13) написать руководство пользователя и администратора программного комплекса.

Предполагаемый функционал проекта:

1) запись информации в базу данных;

2) проверка вводимых данных посредством языков JavaScript и PHP;

3) обновление базы данных;

# 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## 2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для сохранения введённой на сайте информации в базе данных и её изменения (обновления).

## 2.2 Требования к пользовательским интерфейсам

Основные требования к пользовательскому интерфейсу:

1. функциональность;
2. понятность и логичность;
3. обеспечение высокой скорости работы пользователя;
4. обеспечение защиты от человеческих ошибок;
5. быстрое обучение пользователя;

## 2.3 Требования к проектированию программных интерфейсов

При проектировании программных интерфейсов должны учитываться как современные стандарты дизайна, так и требования к конкретному решению.

Основные части функционального элемента «макет сайта-магазина детских игрушек», представленного на рисунке 2.1:

1) логотип магазина;

2) главное меню;

3) раздел «Бестселлер;

4) раздел «Премиум»;

6) раздел «Мягкие игрушки»;

7) раздел «Корзина и оформление заказа».

Рисунок 2.1 – Примерный макет сайта-магазина детских игрушек

Основные части функционального элемента «окно заказа», представленного на рисунке 2.2:

1) название товара;

2) подробное описание;

3) цена;

4) количество;

5) общая сумма;

6) поле для промо-кода;

7) форма для заказа;

8) номер телефона;

9) адрес;

10) кнопка «Заказать».

Рисунок 2.2 – Примерный макет окна заказа

## 2.4 Требования к аппаратным и программным интерфейсам

Для установки и работы с сайтом и его базой данных необходимо иметь вычислительную систему следующей базовой конфигурации:

1) процессор: 1.8 ГГц;

2) оперативная память: 1 Гб;

3) доступ к интернету;

4) до 900 Мб свободной памяти.

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. браузер с поддержкой JavaScript;
3. локальный сервер (XAMPP или аналоги);
4. интерпретатор скриптов PHP;
5. программу для работы с MySQL;
6. установленный Open Server.

## 2.5 Требования к пользователям продукта

Для работы с сайтом пользователь должен иметь:

1) браузер с поддержкой JavaScript;

2) доступ в Интернет;

3) устройства ввода-вывода;

4) базовые навыки работы с компьютером и в Интернете.

## 2.6 Функции продукта

1) Сохранение данных в БД;

2) изменение данных в БД.

## 2.7 Ограничения

1. Сайт не будет корректно работать при отсутствии в пользовательском браузере поддержки JavaScript;
2. Сайт не будет работать при отсутствии подключения к сети Интернет.

# 3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ

## 3.1 Язык программирования серверных скриптов PHP

Скриптовый язык PHP (от английского PHP: Hypertext Preprocessor — «PHP: препроцессор гипертекста»; первоначально Personal Home Page Tools [9]) был разработан специально для веб-программирования. Вобрав в себя достоинства таких языков как C и Perl, он считается достаточно простым в изучении и обладает рядом преимуществ.

Первое его воплощение было создано в 1994 году Расмусом Лердорфом, и являло собой простой набор CGI-скриптов, написанных на языке программирования Си, которые автор использовал для личных целей. Со временем Расмусу потребовалось все больше и больше улучшений и расширения функциональности, и он переписал PHP Tools, добавив более богатую реализацию. Теперь появилась возможность работать с базами данных и создавать динамические приложения.

Существенное преобразование произошло в апреле 1996 года. Расмус представил PHP/FI. Именно тогда началась разработка PHP как языка программирования, а не набора инструментов. Теперь была доступна поддержки DBM, mSQL и Postgres95 баз данных, cookies, возможности исользования определяемых пользователем функций и многое другое. Но те ранние наработки были мало похожи на тот язык PHP, который знают сегодня.

PHP 3.0, который был официально представлен в июне 1998 года, был первой версией, напоминающей современный PHP. Энди Гутманс и Зив Сураски переписали парсер и изменили имя на короткое «PHP».

На данный момент используется PHP 7, большая версия языка с множественными обновлениями, разработанная в течение 2014 и 2015 годов.

Существуют много языков для веб-программирования, но PHP является одним из самых популярных. В чем же его достоинства?

1) Быстрый: использую большее количество системных ресурсов, он быстрее многих других языков;

2) легкий в использовании и изучении: для тех, кто знаком с С и Perl, легко начать работать с PHP благодаря схожести синтаксиса, а отсутствие строгой типизации, такой как в Java или C++, делает этот язык простым в изучении;

3) стабильный: PHP существует уже около двадцати четырех лет. В течение этой эры многие разработчики работали над приложением для улучшения его использования. На протяжении многих лет было обнаружено множество ошибок, и баги исправляются быстро командой разработчиков. Исходя из этого, язык программирования в настоящее время очень стабилен.

4) встроенные модули связи с базами данных (более 20 видов БД): многие сайты используют базы данных, а значит подобные модули – необходимость;

На данный момент, PHP самый популярный серверный язык программирования. На рисунке 3.1 можно увидеть статистику использования серверных языков программирования для веб-сайтов на апрель 2020 года по данным W3Techs – организации, занимающейся сбором данных о веб-технологиях [5].

Рисунок 3.1 – Использование серверных языков

## 3.2 Сервер баз данных MYSQL

Основы современной информационной технологии составляют базы данных. MySQL — свободная реляционная система управления базами данных. Первый внутренний выпуск MySQL состоялся 23 мая 1995 года.

База данных MySQL — это самая популярная в мире база данных с открытым кодом. Благодаря своей проверенной производительности, надежности и простоте использования база данных MySQL наиболее часто используется для веб-приложений на таких ресурсах, как Facebook, Twitter, YouTube и все пять из пяти лучших веб-сайтов. Кроме того, ее особенно часто выбирают в качестве встроенной базы данных, распространяемой тысячами поставщиков программного обеспечения и производителей оборудования.

Преимущества использования MySQL:

1) Безопасность данных

MySQL известна во всем мире как самая безопасная и надежная система управления базами данных, используемая в популярных веб-приложениях, таких как WordPress, Drupal, Joomla, Facebook и других. Защита данных и поддержка транзакционной обработки, которые сопровождают недавнюю версию MySQL, могут значительно принести пользу любому бизнесу, особенно если это бизнес электронной коммерции, который включает частые денежные переводы.

2) Масштабируемость по требованию

MySQL предлагает непревзойденную масштабируемость для облегчения управления глубоко внедренными приложениями, используя меньшую площадь.

3) Высокая производительность

В MySQL есть отдельная инфраструктура хранилища, которая позволяет системным администраторам настраивать сервер базы данных MySQL для безупречной производительности. Будь это веб-сайт электронной коммерции, который получает миллион запросов каждый день или высокоскоростная транзакционная система обработки, MySQL предназначен для удовлетворения даже самых требовательных приложений, обеспечивая при этом оптимальную скорость, полнотекстовые индексы и уникальные кэши памяти для повышения производительности.

4) Сообщество

Образовалось сообщество людей, которые участвуют как в развитии самого пакета, так и в обучении менее опытных людей работе с ним. Существует огромное количество сайтов и страничек, где можно получить бесплатную и быструю помощь в любое время суток.

## 3.3 WEB-сервер APACHE

Apache HTTP-сервер — свободный веб-сервер. Apache сыграл ключевую роль в начальном росте Интернета, быстро обогнав NCSA HTTPd в качестве доминирующего HTTP-сервера и оставаясь наиболее популярным с апреля 1996 года. В 2009 году он стал первым программным обеспечением для веб-серверов, которое обслуживало больше чем 100 миллионов сайтов. По данным Netcraft, в апреле 2020 года более 450 миллионов сайтов использовали Apache HTTP server [1].

Самая простая функция, которую может выполнять Apache – стоять на сервере и обслуживать обычный HTML-сайт. При получении запроса на определенную страницу сервер отправляет в ее ответ браузеру. Самый простой пример: открытие сайта после набора адреса.

Возможность интеграции PHP в Apache предусмотрена разработчиками самого PHP. Apache же выполняет только функции посредника между скриптом и компилятором. Есть два способа интеграции PHP в Apache. Первый – установка специального модуля, расширяющего возможности сервера, и тогда он сам становится способным "компилировать" скрипты. И второй – установка в конфигурационных файлах связей между php-файлами и самим компилятором.

## 3.4 Набор инструментов Bootstrap

Bootstrap (или Twitter Bootstrap) —  это свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. Изначально Bootstrap был создан разработчиками Твиттера для внутреннего пользования, но спустя время он был выложен в открытый доступ.

Преимущества языка заключаются в следующем:

1) адаптивный веб-дизайн. С каждым годом использование мобильных устройств продолжает расти, и их популярность рождает потребность в быстродействии. Создание сайта, готового для мобильных устройств, является быстрой и простой задачей с помощью Bootstrap. Сайты, созданные на Bootstrap’е, идентично отображаются на разных устройствах и в современных браузерах;

2) скорость разработки сайта. Благодаря большому количеству готовых решений и шаблонов, создать каркас сайта – дело пяти минут;

3) легкость в использовании и освоении;

4) одинаковый результат работы независимо от браузера;

5) кастомизация: Bootstrap позволяет настроить все свои компоненты так, как то требуется разработчику.

## 3.5 Язык программирования JAVASCRIPT

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

JavaScript изначально создавался для того, чтобы оживить веб-странички и добавить им интерактивности. Программы на JavaScript скриптами. В браузере они подключаются напрямую к HTML и, как только загружается страничка – тут же выполняются.

JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется. Это сделано для того, чтобы обезопасить пользователей. С помощью JavaScript нельзя получить личные данные с пользовательского компьютера и каким-либо образом навредить ему.

Преимущества использования JavaScript в веб-разрабтке:

1) Javascript выполняется на стороне клиента. Это означает, что код выполняется на процессоре пользователя, а не на веб-сервере, что позволяет уменьшить нагрузку на веб-сервер.

2) Javascript – относительно простой язык

Язык Javascript относительно прост в изучении и содержит синтаксис, близкий к английскому языку. Он использует модель DOM, которая обеспечивает множество предустановленных функций для различных объектов на страницах, что делает его легким для разработки конкретного решения пользовательской цели.

3) Javascript относительно быстрый для конечного пользователя

Поскольку код выполняется на компьютере пользователя, результаты и обработка завершаются почти мгновенно в зависимости от задачи (задачи в javascript на веб-страницах обычно создаются простыми, чтобы не занимать памятью), поскольку ее не нужно обрабатывать на веб-сервере сайта и она отправляется обратно пользователю.

4) Расширенная функциональность для веб-страниц

Сторонние дополнения, такие как Greasemonkey, позволяют разработчикам Javascript писать фрагменты кода JS, которые могут выполняться на желаемых веб-страницах для расширения ее функциональности. Если вы используете веб-сайт и хотите создать определенную функцию, вы можете написать ее самостоятельно и использовать надстройку браузера, такую ​​как Greasemonkey, для ее реализации на веб-странице.

5) Популярность

Благодаря популярности JavaScript не составляет сложности найти сайты, посвященные изучению этого языка и решению тех проблем и задач, которые могут возникнуть во время его использования. Также его популярность стала причиной появления множества фреймворков, которые существенно облегчат работу. Благодаря ним проекты, на написание которых раньше уходили месяцы и сотни строк кода, теперь могут быть написаны гораздо быстрее с хорошо структурированными заранее созданными шаблонами и функциями. Большинство фреймворков являются бесплатными и имеют открытый исходный код.

Есть как минимум три главных и определяющих использование JavaScript в проекте преимущества перед другими языками подобного типа:

1) Полная интеграция с HTML/CSS.

2) Простые вещи делаются просто.

3) Поддерживается всеми распространёнными браузерами и включён по умолчанию.

Этих трёх вещей одновременно нет больше ни в одной браузерной технологии. Поэтому JavaScript и является самым распространённым средством создания браузерных интерфейсов.

## 3.6 Серверная и программная платформа Open Server

Open Server Panel — это портативная серверная платформа и программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий [5].

Программный комплекс имеет богатый набор серверного программного обеспечения, удобный, многофункциональный продуманный интерфейс, обладает мощными возможностями по администрированию и настройке компонентов. Платформа широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях [5].

Хотя изначально программные продукты, входящие в состав комплекса, не разрабатывались специально для работы друг с другом, такая связка стала весьма популярной среди пользователей Windows, в первую очередь из-за того, что они получали бесплатный комплекс программ с надежностью на уровне Linux серверов [5].

Удобство и простота управления безусловно не оставят вас равнодушными, за время своего существования Open Server зарекомендовал себя как первоклассный и надёжный инструмент необходимый каждому веб-мастеру [5].

Идея проекта Open Server заключается в независимости от рабочего места. Обычный разработчик часто зависит от конкретного компьютера, от операционной системы и программ установленных на этом компьютере, и даже от дома или офиса где находится этот компьютер. Open Server призван освободить вас и ваши программы от подобных неудобств, он сделает вас по-настоящему свободным и независимым разработчиком [5].

## 3.7 Архитектура «клиент-сервер»

**Архитектура клиент/сервер** — **это архитектура производителей / потребителей, где сервер выступает в роли производителя и клиента как потребителя. Сервер обслуживает клиентов и предоставляет клиентам высокопроизводительные вычислительные ресурсы. Эти службы могут включать доступ к приложениям, хранение, совместное использование файлов, доступ к принтеру и / или прямой доступ к исходной вычислительной мощности сервера.**

**Архитектура клиент/сервер работает, когда клиентский компьютер отправляет ресурс или запрос процесса серверу по сетевому соединению, который затем обрабатывается и доставляется клиенту. Серверный компьютер может управлять несколькими клиентами одновременно, тогда как один клиент может быть подключен к нескольким серверам за раз, каждый из которых предоставляет другой набор услуг. В своей простейшей форме Интернет также основан на архитектуре клиент/сервер, где веб-серверы обслуживают множество одновременных пользователей с данными веб-сайта.**

Пример: Когда клиент банка обращается к онлайн-банковским услугам с помощью веб-браузера (клиента), клиент инициирует запрос на веб-сервер банка. Учетные данные пользователя могут храниться в базе данных, а веб-сервер обращается к серверу базы данных как к клиенту. Сервер приложений интерпретирует возвращенные данные, применяя бизнес-логику банка и предоставляет вывод на веб-сервер. Наконец, веб-сервер возвращает результат в клиентский веб-браузер для отображения.

# 4 РАЗРАБОТКА БАЗЫ ДАННЫХ САЙТА МАГАЗИНА ПО ПРОДАЖЕ ДЕТСКИХ ИГРУШЕК

## 4.1 Выбор и обоснование CУБД

СУБД — это приложения (или библиотеки), управляющие базами данных различных форм, размеров и типов.

Реляционные системы (РСУБД) реализуют реляционную модель работы с данными, которая определяет всю хранимую информацию как набор связанных записей и атрибутов в таблице.

СУБД такого типа используют структуры (таблицы) для хранения и работы с данными. Каждый столбец (атрибут) содержит свой тип информации. Каждая запись в базе данных, обладающая уникальным ключом, передаётся в строку таблицы, и её атрибуты отображаются в столбцах таблицы.

Таблица 4.1 – Описание различных СУБД

|  |  |  |
| --- | --- | --- |
| Название | Достоинства | Недостатки |
| MySQL | Самая популярная и часто используемая РСУБД;  Много функций: MySQL поддерживает большую часть функционала SQL.  Безопасность: в MySQL встроено много функций безопасности.  Мощность и масштабируемость: MySQL может работать с действительно большими объёмами данных, и неплохо походит для масштабируемых приложений.  Скорость: пренебрежение некоторыми стандартами позволяет MySQL работать производительнее, местами срезая на поворотах.  Быстродействие. Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко. | Известные ограничения: присутствуют определённые ограничения функциональности.  Вопросы надёжности: некоторые операции реализованы менее надёжно, чем в других РСУБД.  Отсутствие полной совместимости с SQL, что может вызвать проблемы при переносе БД. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| PostgreSQL | Самая продвинутая и гибкая РСУБД.  Полная SQL-совместимость.  Расширяемость: PostgreSQL можно программно расширить за счёт хранимых процедур.  Объектно-ориентированность: PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД. | Производительность: В простых операциях чтения PostgreSQL может уступать своим соперникам.  Популярность: из-за своей сложности инструмент не очень популярен.  Скорость: если всё, что нужно — это быстрые операции чтения, не стоит использовать PostgreSQL. |
| SQLite | Очень мощная встраиваемая РСУБД.  Вся база данных хранится в одном файле, что облегчает перемещение.  Отлично подходит для разработки и даже тестирования: во время этапа разработки большинству требуется масштабируемое решение. SQLite, со своим богатым набором функций, может предоставить более чем достаточный функционал, при этом будучи достаточно простой для работы с одним файлом и связанной сишной библиотекой. | Отсутствие пользовательского управления: продвинутые БД предоставляют пользователям возможность управлять связями в таблицах в соответствии с привилегиями, но у SQLite такой функции нет.  Невозможность дополнительной настройки: SQLite нельзя сделать более производительной, изменив настройки. |

На основании полученных данных и их сравнения, для работы в данном проекте была выбрана реляционная система MySQL.

## 4.2 Концептуальное проектирование БД

Назначение концептуальной модели состоит в том, чтобы представить формализованную информацию о предметной области таким образом, чтобы она было достаточно емкой для оценки глубины и корректности проработки проекта базы данных.

На схеме, представленной ниже, были выделены следующие сущности, необходимые для решения задачи:

1. пользователь;  
2. клиент;  
3. сообщение;  
4. заказ;  
5. главная страница сайта;  
6. карточки товара;  
7. окно заказа товара;  
8. администратор;

Рисунок 4.1 – Логическая модель базы данных

## 4.3 Физическое проектирование БД

Рисунок 4.2 – Физическая модель базы данных

Физическое проектирование базы данных подразумевает процесс подготовки переноса реализации базы данных на определенную СУБД. Для построения данной схемы было выбрано приложение MySQL Workbench.

Физическая модель данных описывает реализацию объектов логической модели на уровне объектов конкретной базы данных.

База данных проекта состоит из 8 таблиц:

1. users;
2. event-group;
3. event-types;
4. events;
5. event-statusers;
6. orders;
7. items;
8. order-items.

Структура каждой из таблиц и типы данных внутри таблицы представлены на рисунке 4.1.

# 5 САЙТ МАГАЗИН ПО ПРОДАЖЕ ДЕТСКИХ ИГРУШЕК

## 5.1 Входные и выходные данные

Входные данные сайта – данные пользователя.

Результатом работы приложения является запись в базу данных.

## 5.2 Структура сайта

На схеме 5.1 показана структура файлов сайта.

├───курсовая

├───admin

├───assets

├───cron

├───hostcmsfiles

├───images

│ └───user

├───modules

│ ├───admin

│ ├───benchmark

│ ├───calendar

│ ├───comment

│ ├───company

│ ├───constant

│ ├───core

│ ├───crm

│ ├───directory

│ ├───document

│ ├───event

│ ├───informationsystem

│ ├───lib

│ ├───market

│ ├───property

│ ├───shop

│ ├───shortcode

│ ├───source

│ │ └───i18n

│ ├───sql

│ │ ├───dataset

│ │ └───i18n

│ ├───structure

│ │ ├───controller

│ │ │ └───edit

│ │ ├───i18n

│ │ ├───menu

│ │ │ ├───controller

│ │ │ └───i18n

│ │ └───property

│ │ ├───dir

│ │ └───list

│ ├───tag

│ │ ├───controller

│ │ ├───dir

│ │ │ └───i18n

│ │ ├───i18n

│ │ ├───informationsystem

│ │ │ └───item

│ │ │ └───i18n

│ │ └───shop

│ │ └───item

│ │ └───i18n

│ ├───template

│ │ ├───controller

│ │ ├───dir

│ │ │ └───i18n

│ │ ├───i18n

│ │ ├───less

│ │ │ └───command

│ │ ├───preprocessor

│ │ └───section

│ ├───tpl

├───templates

└───upload

Рисунок 5.1 – Структура файлов

## 5.3 Верстка сайта

Для создания сайта использовался фреймворк Bootstrap, который был подробно описан ранее. Он предоставляет набор интрументов для создания сайтов и веб-страниц. Данный фреймворк включает в себя HTML и CSS шаблоны, а также готовые расширения Javascript.

Bootstrap позволяет создать адаптивный сайт с помощью резиновой системы разметки, которая масштабируется до 12 столбцов на различных устройсвах. Количество столбцов можно менять при надобновсти. Подобный масштаб позволяет создавать как простые варианты разметки, так и более сложные макеты.

Система разметки Bootstrap также имеет собственные правила размещения элементов, которые рассмотрим ниже:

1) элементы .row всегда помещаются в .container (с фиксированной шириной) или в .container-fluid (полная ширина) для корректного выравнивания и наполнения;

2) строки используются для создания горизонтальной группы столбцов;

3) в столбцах помещается контент, только колонки могут быть непосредственными потомками строк;

4) для быстрого создания сетки макетов используются стандартные классы сетки (например .row или .col-xs-4).

## 5.4 Описание алгоритма работы скриптов и SQL-запросов сайта

Соединение с сервером базы данных, создание запроса для вывода

public function \_\_construct(array $args = array())

{

// Add columns for select

call\_user\_func\_array(array($this, 'columns'), $args);

return parent::\_\_construct($args);

}

/\*\*

\* Add columns for select

\*

\* <code>

\* // SELECT `id`, `name`

\* $Core\_QueryBuilder\_Select->columns('id', 'name');

\*

\* // SELECT `id`, `name` AS `aliasName`

\* $Core\_QueryBuilder\_Select->columns('id', array('name', 'aliasName'));

\*

\* // SELECT `id`, `tablename`.`name` AS `aliasName`

\* $Core\_QueryBuilder\_Select->columns('id', array('tablename.name', 'aliasName'));

\* </code>

\* @return Core\_QueryBuilder\_Select

\*/

public function columns()

{

$args = func\_get\_args();

$this->\_select = array\_merge($this->\_select, $args);

return $this;

}

/\*\*

\* Alias for columns()

\*

\* @see columns()

\*/

public function select()

{

$args = func\_get\_args();

return call\_user\_func\_array(array($this, 'columns'), $args);

}

/\*\*

\* Set HIGH\_PRIORITY

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->highPriority();

\* </code>

\* @return Core\_QueryBuilder\_Select

\*/

public function highPriority()

{

$this->\_highPriority = TRUE;

return $this;

}

/\*\*

\* Set STRAIGHT\_JOIN

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->straightJoin();

\* </code>

\* @return Core\_QueryBuilder\_Select

\*/

public function straightJoin()

{

$this->\_straightJoin = TRUE;

return $this;

}

/\*\*

\* Set SQL\_CALC\_FOUND\_ROWS

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->sqlCalcFoundRows();

\* </code>

\* @param boolean $value

\* @return Core\_QueryBuilder\_Select

\*/

public function sqlCalcFoundRows($value = TRUE)

{

$this->\_sqlCalcFoundRows = $value ? TRUE : NULL;

return $this;

}

/\*\*

\* Set DISTINCT and add columns for select

\*

\* @see columns()

\* @return Core\_QueryBuilder\_Select

\*/

public function selectDistinct()

{

$args = func\_get\_args();

$this->distinct()->columns($args);

return $this;

}

/\*\*

\* Set DISTINCT

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->distinct();

\* </code>

\* @param boolean $distinct DISTINCT option

\* @return Core\_QueryBuilder\_Select

\*/

public function distinct($distinct = TRUE)

{

$this->\_distinct = $distinct;

return $this;

}

Полный код приведён в Приложении Б Листинг Б.1 **–** Подсоединение к базе данных, модель для создания запроса для вывода товара.

Запись информации о сессии в базу данных

class Core\_Session\_Database extends Core\_Session

{

/\*\*

\* DataBase instance

\* @var Core\_DataBase

\*/

protected $\_dataBase = NULL;

/\*\*

\* Session has been read

\* @var boolean

\*/

protected $\_read = FALSE;

/\*\*

\* Lock prefix

\* @var string

\*/

protected $\_lockPrefix = NULL;

/\*\*

\* GET\_LOCK timeout (sec)

\* @var int

\*/

protected $\_getLockTimeout = 5;

/\*\*

\* Next step delay (microseconds)

\* Default 0,5 sec - 500000 microseconds

\* @var int

\*/

protected $\_nextStepDelay = 500000;

/\*\*

\* Lock timeout

\* @var int

\*/

protected $\_lockTimeout = 10;

/\*\*

\* Constructor.

\*/

public function \_\_construct()

{

$this->\_dataBase = Core\_DataBase::instance();

if (is\_null($this->\_lockPrefix))

{

$aDataBaseConfig = $this->\_dataBase->getConfig();

$this->\_lockPrefix = $aDataBaseConfig['database'] . '\_' . 'sessions';

}

}

/\*\*

\* The open callback works like a constructor in classes and is executed when the session is being opened.

\* @param string $save\_path save path

\* @param string $session\_name session name

\* @return boolean

\*/

public function sessionOpen($save\_path, $session\_name)

{

return TRUE;

}

/\*\*

\* The close callback works like a destructor in classes and is executed after the session write callback has been called.

\* @return boolean

\*/

public function sessionClose()

{

return TRUE;

}

Полный код приведён в Приложении Б Листинг Б.2– Обработка отдельной сессии, запись сессии в базу данных

# 6 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 6.1 Аппаратные, системные и программные требования

Для просмотра сайта необходимо иметь вычислительную систему следующей базовой конфигурации:

1) процессор: 1.8 ГГц;

2) оперативная память: 1 Гб;

3) доступ к интернету.

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. браузер с поддержкой JavaScript.

Для работы с кодом сайта и его базой данных необходимо иметь вычислительную систему следующей базовой конфигурации:

1) процессор: 1.8 ГГц;

2) оперативная память: 1 Гб;

3) доступ к интернету;

4) до 900 Мб свободной памяти.

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. браузер с поддержкой JavaScript;
3. локальный сервер (XAMPP или аналоги);
4. интерпретатор скриптов PHP;
5. программу для работы с MySQL;
6. Open Server.

## 6.2 Описание установки и запуска сайта

Для начала работы с сайтом и его базой данных необходимо установить локальный сервер, например XAMPP, который идёт с Open Server, MySQL, интерпретатор скриптов PHP, язык программирования jQuery и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер. Установка может быть как через стандартный установщик Windows, так и с помощью распаковки архивов zip или 7zip.

Следующий шаг – запуск Open Server и MySQL. Для этого необходимо открыть папку с программой и запустить Open Server x64, либо Open Server x86, как изображено на рисунке 6.1.

Рисунок 6.1 – Запуск Open Server

Программа запускается автоматически в фоновом режиме. Чтобы проверить работу сервера, необходимо ввести в браузер http://localhost/ или http://127.0.0.1/. Выйдет начальная страница Open Server.

Для того, чтобы запустить файлы сайта, например, index.html, их нужно переместить/скопировать по адресу [нет особых предпочтений, где храниться, можно даже на флешке]\OSPanel\domeins\[ваша папка с проектом].

Рисунок 6.2 – Путь для копирования файлов для запуска на localhost

Вместо диска С может быть другая директория. После этого ввести в браузере index/.

Рисунок 6.3 – Открытый сайт на localhost

### 6.3 Установка, настройка и запуск программы для работы с базами данных

Для работы с базой данных были выбраны два решения: сайт GearHost для хранения данных и программа MySQL Workbench для работы с БД. Преимущество GearHost заключается в бесплатном варианте его использования, но он ограничен – 5 МБ всего. MySQL Workbench может выполнить различные задания, среди которых дизайн и моделирование баз данных, работа с SQL, управление базой данных, миграция базы данных.

Для создания базы данных на GearHost необходимо зарегистрироваться на нем и в разделе «[Databases](https://my.gearhost.com/Databases)» создать базу данных. Автоматически будет создан пользователь базы данных. Данные необходимо будет ввести в MySQL Workbench.

Для работы с программой MySQL Workbench ее необходимо скачать с официального сайта MySQL и установить. После установки открыть, нажать Database > Connect to Database… или Ctrl + U, как изображено на следующем рисунке:

Рисунок 6.4 – Подключение к БД в MySQL Workbench

Откроется окно подключения к базе данных, куда необходимо ввести данные с сайта GearHost таким образом: Database server – Hostname, Username – shop:

Рисунок 6.5 – Ввод данных в MySQL Workbench

После этого необходимо ввести пароль и работать с базой данных.

Рисунок 6.6 – Главное окно работы MySQL Workbench

# ЗАКЛЮЧЕНИЕ

В результате разработки сайта магазина по продаже детских игрушек был проведён анализ аналогов на рынке, также разработан и реализован конкурентно способный аналог.

Была исследована документация языка гипертекстовой разметки HTML, фреймворка Bootstrap, скриптового языка PHP и языка программирования JavaScript, как и возможности реляционной системы управления базами данных MySQL.

Язык PHP был проанализирован с точки зрения работы с базой данных через сайт, была оценена работа JavaScript в валидации вводимых данных, а Bootstrap в создания адаптивных сайтов.

В ходе работы была создана адаптивная страница сайта с описанием сервиса и подробной информации о нем. Кроме того, была создана страница заполнения заказа с записью в базу данных.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. April 2018 Web Server Survey // Netcraft – Web Server Survey URL: https://news.netcraft.com/archives/category/web-server-survey/ (дата обращения: 12.03.2020)
2. STATUS QUO Сравнение украинских интернет-магазинов игрушек Беби Хеппи и Панама URL: https://www.sq.com.ua/rus/news/novosti\_partnerov/02.03.2020/sravnenie\_ukrainskih\_internet\_magazinov\_igrushek\_bebi\_heppi\_i\_panama/ (дата обращения: 12.03.2020)
3. PHP: History of PHP - Manual // PHP: Hypertext Preprocessor URL: http://php.net/manual/en/history.php.php (дата обращения: 06.03.2020)
4. Usage Statistics and Market Share of Server-side Programming Languages for Websites, April 2018 // W3Techs URL: https://w3techs.com/technologies/overview/programming\_language/all (дата обращения: 06.03.2020)
5. Встречайте: Open Server! URL: https://ospanel.io/ (дата обращения: 26.03.2020)
6. ГДЕ ВЫГОДНЕЕ КУПИТЬ ИГРУШКИ? URL: https://essatoys.ua/kypit-igrushki-optom/gde-vygodnee-kupit-igrushki (дата обращения: 12.03.2020)
7. Зачем нужен интернет-магазин? URL: https://webevolution.ru/blog/sajti/zachem-nuzhen-internet-magazin/ (дата обращения: 12.03.2020)
8. ПЛЮШЕВЫЕ ИГРУШКИ – МИЛОЕ ОЧАРОВАНИЕ! Roses URL: https://igrushkioptom.com.ua/pljushevye-igrushki-miloe-ocharovanie/ (дата обращения: 12.03.2020)
9. Репин В. В. Процессный подход к управлению. Моделирование бизнес-процессов [Текст] // В. В. Репин, В. Г. Епиферов. – 2-е изд. – М.: РИА «Стандарты и качество», 2005. – 404 с

# ПРИЛОЖЕНИЕ А

**Экранные формы**

Рисунок А.1 – Главная страница сайта (Бестселлеры)

Рисунок А.2 – Главная страница сайта (Премиум)

Рисунок А.3 – Главная страница сайта (Мягкие игрушки)

Рисунок А.4 – Главная страница сайта (Корзина)

Рисунок А.5 – Заказ оформлен

# ПРИЛОЖЕНИЕ Б

**Фрагменты листинга**

Листинг Б.1 **–** Подсоединение к базе данных, модель для создания запроса для вывода товара

class Core\_QueryBuilder\_Select extends Core\_QueryBuilder\_Selection

{

/\*\*

\* DISTINCT

\* @var bool

\*/

protected $\_distinct = FALSE;

/\*\*

\* HIGH\_PRIORITY

\* @var mixed

\*/

protected $\_highPriority = NULL;

/\*\*

\* STRAIGHT\_JOIN

\* @var mixed

\*/

protected $\_straightJoin = NULL;

/\*\*

\* SQL\_CALC\_FOUND\_ROWS

\* @var mixed

\*/

protected $\_sqlCalcFoundRows = NULL;

/\*\*

\* SELECT

\* @var array

\*/

protected $\_select = array();

/\*\*

\* FROM

\* @var array

\*/

protected $\_from = array();

/\*\*

\* GROUP BY

\* @var array

\*/

protected $\_groupBy = array();

/\*\*

\* HAVING

\* @var array

\*/

protected $\_having = array();

/\*\*

\* Array of Core\_QueryBuilder\_Select's

\* @var array

\*/

protected $\_union = array();

/\*\*

\* UNION ORDER BY

\* @var array

\*/

protected $\_unionOrderBy = array();

/\*\*

\* UNION LIMIT

\* @var mixed

\*/

protected $\_unionLimit = NULL;

/\*\*

\* UNION OFFSET

\* @var mixed

\*/

protected $\_unionOffset = NULL;

/\*\*

\* DataBase Query Type

\* 0 - SELECT

\*/

protected $\_queryType = 0;

/\*\*

\* Constructor.

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select('id', array('tableName.name', 'aliasName'));

\* </code>

\*

\* @param array $args list of arguments

\* @see columns()

\*/

public function \_\_construct(array $args = array())

{

// Add columns for select

call\_user\_func\_array(array($this, 'columns'), $args);

return parent::\_\_construct($args);

}

/\*\*

\* Add columns for select

\*

\* <code>

\* // SELECT `id`, `name`

\* $Core\_QueryBuilder\_Select->columns('id', 'name');

\*

\* // SELECT `id`, `name` AS `aliasName`

\* $Core\_QueryBuilder\_Select->columns('id', array('name', 'aliasName'));

\*

\* // SELECT `id`, `tablename`.`name` AS `aliasName`

\* $Core\_QueryBuilder\_Select->columns('id', array('tablename.name', 'aliasName'));

\* </code>

\* @return Core\_QueryBuilder\_Select

\*/

public function columns()

{

$args = func\_get\_args();

$this->\_select = array\_merge($this->\_select, $args);

return $this;

}

/\*\*

\* Alias for columns()

\*

\* @see columns()

\*/

public function select()

{

$args = func\_get\_args();

return call\_user\_func\_array(array($this, 'columns'), $args);

}

/\*\*

\* Set HIGH\_PRIORITY

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->highPriority();

\* </code>

\* @return Core\_QueryBuilder\_Select

\*/

public function highPriority()

{

$this->\_highPriority = TRUE;

return $this;

}

/\*\*

\* Set STRAIGHT\_JOIN

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->straightJoin();

\* </code>

\* @return Core\_QueryBuilder\_Select

\*/

public function straightJoin()

{

$this->\_straightJoin = TRUE;

return $this;

}

/\*\*

\* Set SQL\_CALC\_FOUND\_ROWS

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->sqlCalcFoundRows();

\* </code>

\* @param boolean $value

\* @return Core\_QueryBuilder\_Select

\*/

public function sqlCalcFoundRows($value = TRUE)

{

$this->\_sqlCalcFoundRows = $value ? TRUE : NULL;

return $this;

}

/\*\*

\* Set DISTINCT and add columns for select

\*

\* @see columns()

\* @return Core\_QueryBuilder\_Select

\*/

public function selectDistinct()

{

$args = func\_get\_args();

$this->distinct()->columns($args);

return $this;

}

/\*\*

\* Set DISTINCT

\*

\* <code>

\* $oCore\_QueryBuilder\_Select = Core\_QueryBuilder::select()->distinct();

\* </code>

\* @param boolean $distinct DISTINCT option

\* @return Core\_QueryBuilder\_Select

\*/

public function distinct($distinct = TRUE)

{

$this->\_distinct = $distinct;

return $this;

}

/\*\*

\* Set FROM

\*

\* <code>

\* // FROM `tableName`

\* $oCore\_QueryBuilder\_Select = $Core\_QueryBuilder\_Select->from('tableName');

\*

\* // FROM `tableName` AS `tableNameAlias`

\* $oCore\_QueryBuilder\_Select = $Core\_QueryBuilder\_Select->from(array('tableName', 'tableNameAlias'));

\* </code>

\* @return Core\_QueryBuilder\_Select

\*/

public function from()

{

$args = func\_get\_args();

$this->\_from = array\_merge($this->\_from, $args);

return $this;

}

/\*\*

\* Open bracket in HAVING

\* @return Core\_QueryBuilder\_Select

\*/

public function havingOpen()

{

$this->\_having[] = array(

$this->\_operator => array('(')

);

// Set operator as EMPTY string

$this->\_operator = '';

return $this;

}

/\*\*

\* Close bracket in HAVING

\* @return Core\_QueryBuilder\_Select

\*/

public function havingClose()

{

// Set operator as EMPTY string

$this->\_operator = '';

$this->\_having[] = array(

$this->\_operator => array(')')

);

// Set operator as default

$this->setDefaultOperator();

return $this;

}

/\*\*

\* Add HAVING

\*

\* <code>

\* // HAVING `a1` > '2'

\* $Core\_QueryBuilder\_Select->having('a1', '>', '2');

\*

\* // HAVING `f5` IS TRUE

\* $Core\_QueryBuilder\_Select->having('f5', 'IS', TRUE);

\*

\* // HAVING `a4` IN (17, 19, NULL, 'NULL')

\* $Core\_QueryBuilder\_Select->having('a4', 'IN', array(17,19, NULL, 'NULL'));

\*

\* // HAVING `a7` BETWEEN 1 AND 10

\* $Core\_QueryBuilder\_Select->having('a7', 'BETWEEN', array(1, 10));

\* </code>

\* @param string $column column

\* @param string $expression expression

\* @param string $value value

\* @return Core\_QueryBuilder\_Select

\*/

public function having($column, $expression, $value)

{

$this->\_having[] = array(

$this->\_operator => array($column, $expression, $value)

);

// Set operator as default

$this->setDefaultOperator();

return $this;

}

/\*\*

\* Set GROUP BY

\*

\* <code>

\* // GROUP BY `field`, COUNT(`id`)

\* $Core\_QueryBuilder\_Select->groupBy('field1')->groupBy('COUNT(id)');

\* </code>

\* @param string $column column

\* @return Core\_QueryBuilder\_Select

\*/

public function groupBy($column)

{

$this->\_groupBy[] = $column;

return $this;

}

/\*\*

\* GROUP BY

\* @param string $column column

\* @return Core\_QueryBuilder\_Select

\* @see groupBy()

\*/

public function group($column)

{

return $this->groupBy($column);

}

/\*\*

\* UNION is used to combine the result from multiple SELECT statements into a single result set.

\*

\* <code>

\* // (SELECT `id2`, `name2` FROM `tablename2`)

\* // UNION

\* // (SELECT `id`, `name` FROM `tablename` LIMIT 10 OFFSET 0)

\* $select1 = Core\_QueryBuilder::select('id', 'name')->from('tablename')

\* ->limit(0, 10);

\*

\* $select2 = Core\_QueryBuilder::select('id2', 'name2')->from('tablename2')

\* ->union($select1);

\* </code>

\* @param Core\_QueryBuilder\_Select $object

\* @return Core\_QueryBuilder\_Select

\*/

public function union(Core\_QueryBuilder\_Select $object)

{

$this->\_union[] = array('', $object);

return $this;

}

/\*\*

\* UNION ALL is used to combine the result from multiple SELECT statements into a single result set.

\*

\* <code>

\* // (SELECT `id2`, `name2` FROM `tablename2`)

\* // UNION ALL

\* // (SELECT `id`, `name` FROM `tablename` LIMIT 10 OFFSET 0)

\* $select1 = Core\_QueryBuilder::select('id', 'name')->from('tablename')

\* ->limit(0, 10);

\*

\* $select2 = Core\_QueryBuilder::select('id2', 'name2')->from('tablename2')

\* ->unionAll($select1);

\* </code>

\* @param Core\_QueryBuilder\_Select $object

\* @return Core\_QueryBuilder\_Select

\*/

public function unionAll(Core\_QueryBuilder\_Select $object)

{

$this->\_union[] = array('ALL', $object);

return $this;

}

/\*\*

\* ORDER BY for UNION

\*

\* http://dev.mysql.com/doc/refman/5.7/en/union.html

\* @param string $column column

\* @param string $direction sorting direction

\* @param boolean $binary binary option

\*/

public function unionOrderBy($column, $direction = 'ASC', $binary = FALSE)

{

$direction = strtoupper($direction);

if (in\_array($direction, array('ASC', 'DESC', 'RAND()')))

{

$this->\_unionOrderBy[] = array($column, $direction, $binary);

}

else

{

throw new Core\_Exception("The direction '%direction' doesn't allow",

array('%direction' => $direction)

);

}

return $this;

}

/\*\*

\* LIMIT for UNION

\* @param string $arg1 offset

\* @param string $arg2 count

\*/

public function unionLimit($arg1, $arg2 = NULL)

{

if (!is\_null($arg2))

{

$this->\_unionLimit = intval($arg2);

return $this->unionOffset($arg1);

}

$this->\_unionLimit = intval($arg1);

return $this;

}

/\*\*

\* Set UNION offset

\*

\* @param int $offset offset

\* @return Core\_QueryBuilder\_Selection

\*/

public function unionOffset($offset)

{

$this->\_unionOffset = intval($offset);

return $this;

}

/\*\*

\* Offset compatibility with PostgreSQL, default TRUE

\* @var boolean

\*/

protected $\_offsetPostgreSQLSyntax = TRUE;

/\*\*

\* Offset compatibility with PostgreSQL, default TRUE

\* $param boolean $compatible

\*/

public function offsetPostgreSQLSyntax($compatible)

{

$this->\_offsetPostgreSQLSyntax = $compatible;

return $this;

}

/\*\*

\* Build the SQL query

\*

\* @return string The SQL query

\*/

public function build()

{

$query = array('SELECT');

if ($this->\_distinct)

{

$query[] = 'DISTINCT';

}

if (!is\_null($this->\_highPriority))

{

$query[] = 'HIGH\_PRIORITY';

}

if (!is\_null($this->\_straightJoin))

{

$query[] = 'STRAIGHT\_JOIN';

}

if (!is\_null($this->\_sqlCalcFoundRows))

{

$query[] = 'SQL\_CALC\_FOUND\_ROWS';

}

if (!empty($this->\_select))

{

$query[] = implode(', ', $this->\_quoteColumns($this->\_select));

}

else

{

$query[] = '\*';

}

if (!empty($this->\_from))

{

$query[] = 'FROM ' . implode(', ', $this->\_quoteTables($this->\_from));

}

if (!empty($this->\_join))

{

$query[] = $this->\_buildJoin($this->\_join);

}

if (!empty($this->\_where))

{

$query[] = 'WHERE ' . $this->\_buildExpression($this->\_where);

}

if (!empty($this->\_groupBy))

{

$query[] = 'GROUP BY ' . implode(', ', $this->\_quoteColumns($this->\_groupBy));

}

if (!empty($this->\_having))

{

$query[] = 'HAVING ' . $this->\_buildExpression($this->\_having);

}

if (!empty($this->\_orderBy))

{

$query[] = $this->\_buildOrderBy($this->\_orderBy);

}

if (!is\_null($this->\_limit))

{

$sLImit = 'LIMIT ';

!$this->\_offsetPostgreSQLSyntax && !is\_null($this->\_offset) && $sLImit .= $this->\_offset . ', ';

$query[] = $sLImit . $this->\_limit;

}

if ($this->\_offsetPostgreSQLSyntax && !is\_null($this->\_offset))

{

$query[] = 'OFFSET ' . $this->\_offset;

}

$sql = implode(" \n", $query);

// Unions

if (!empty($this->\_union))

{

$aUnion = array('(', $sql);

foreach ($this->\_union as $aTmpUnion)

{

list($unionType, $oUnion) = $aTmpUnion;

$aUnion[] = ") \nUNION {$unionType}\n(";

$aUnion[] = $oUnion->build();

}

$aUnion[] = ')';

if (!empty($this->\_unionOrderBy))

{

$aUnion[] = ' ' . $this->\_buildOrderBy($this->\_unionOrderBy);

}

if (!is\_null($this->\_unionLimit))

{

$aUnion[] = ' LIMIT ' . $this->\_unionLimit;

}

if (!is\_null($this->\_unionOffset))

{

$aUnion[] = ' OFFSET ' . $this->\_unionOffset;

}

$sql = implode('', $aUnion);

}

return $sql;

}

/\*\*

\* Clear HAVING list

\* @return Core\_QueryBuilder\_Select

\*/

public function clearHaving()

{

$this->\_having = array();

return $this;

}

/\*\*

\* Get SELECT

\* @return array

\*/

public function getSelect()

{

return $this->\_select;

}

/\*\*

\* Clear SELECT list

\* @return Core\_QueryBuilder\_Select

\*/

public function clearSelect()

{

$this->\_select = array();

return $this;

}

/\*\*

\* Clear FROM

\* @return Core\_QueryBuilder\_Select

\*/

public function clearFrom()

{

$this->\_from = array();

return $this;

}

/\*\*

\* Clear GROUP BY list

\* @return Core\_QueryBuilder\_Select

\*/

public function clearGroupBy()

{

$this->\_groupBy = array();

return $this;

}

/\*\*

\* Clear

\* @return Core\_QueryBuilder\_Select

\*/

public function clear()

{

$this->\_distinct = $this->\_unbuffered = FALSE;

$this->\_unionLimit = $this->\_unionOffset = $this->\_highPriority = $this->\_straightJoin

= $this->\_sqlCalcFoundRows = NULL;

$this->\_select = $this->\_from

= $this->\_join = $this->\_groupBy

= $this->\_having = $this->\_union = array();

return parent::clear();

}

}

Листинг Б.2 – Обработка отдельной сессии, запись сессии в базу данных

class Core\_Session\_Database extends Core\_Session

{

/\*\*

\* DataBase instance

\* @var Core\_DataBase

\*/

protected $\_dataBase = NULL;

/\*\*

\* Session has been read

\* @var boolean

\*/

protected $\_read = FALSE;

/\*\*

\* Lock prefix

\* @var string

\*/

protected $\_lockPrefix = NULL;

/\*\*

\* GET\_LOCK timeout (sec)

\* @var int

\*/

protected $\_getLockTimeout = 5;

/\*\*

\* Next step delay (microseconds)

\* Default 0,5 sec - 500000 microseconds

\* @var int

\*/

protected $\_nextStepDelay = 500000;

/\*\*

\* Lock timeout

\* @var int

\*/

protected $\_lockTimeout = 10;

/\*\*

\* Constructor.

\*/

public function \_\_construct()

{

$this->\_dataBase = Core\_DataBase::instance();

if (is\_null($this->\_lockPrefix))

{

$aDataBaseConfig = $this->\_dataBase->getConfig();

$this->\_lockPrefix = $aDataBaseConfig['database'] . '\_' . 'sessions';

}

}

/\*\*

\* The open callback works like a constructor in classes and is executed when the session is being opened.

\* @param string $save\_path save path

\* @param string $session\_name session name

\* @return boolean

\*/

public function sessionOpen($save\_path, $session\_name)

{

return TRUE;

}

/\*\*

\* The close callback works like a destructor in classes and is executed after the session write callback has been called.

\* @return boolean

\*/

public function sessionClose()

{

return TRUE;

}

/\*\*

\* The read callback must always return a session encoded (serialized) string, or an empty string if there is no data to read.

\* @param string $id session ID

\* @return string

\*/

public function sessionRead($id)

{

if ($this->\_lock($id))

{

$queryBuilder = Core\_QueryBuilder::select('value')

->from('sessions')

->where('id', '=', $id)

->limit(1);

$row = $queryBuilder->execute()->asAssoc()->current();

$this->\_read = TRUE;

if ($row)

{

// Update last change time

Core\_QueryBuilder::update('sessions')

//->columns(array('time' => 'UNIX\_TIMESTAMP(NOW())'))

->columns(array('time' => time()))

->where('id', '=', $id)

->execute();

return base64\_decode($row['value']);

}

}

return '';

}

/\*\*

\* The write callback is called when the session needs to be saved and closed.

\* @param string $id session ID

\* @param string $value data

\* @return boolean

\*/

public function sessionWrite($id, $value)

{

if ($this->\_read/\* && $this->\_lock($id)\*/)

{

$value = base64\_encode($value);

$oDataBase = Core\_QueryBuilder::update('sessions')

//->columns(array('time' => 'UNIX\_TIMESTAMP(NOW())'))

->set('value', $value)

->set('time', time())

->where('id', '=', $id)

->execute();

// Returns the number of rows affected by the last SQL statement

// If nothing's really was changed affected rowCount will return 0.

if ($oDataBase->getAffectedRows() == 0 && $value != '')

{

$maxlifetime = self::getMaxLifeTime();

Core\_QueryBuilder::insert('sessions')

->ignore()

->columns('id', 'value', 'time', 'maxlifetime')

->values($id, $value, time(), $maxlifetime)

->execute();

}

$this->\_unlock($id);

$this->\_read = FALSE;

}

return TRUE;

}

/\*\*

\* This callback is executed when a session is destroyed with session\_destroy()

\* @param string $id session ID

\* @return boolean

\*/

public function sessionDestroyer($id)

{

if ($this->\_lock($id))

{

Core\_QueryBuilder::delete('sessions')

->where('id', '=', $id)

->execute();

// для предотвращения автоматической повторной регистрации сеанса

$\_SESSION = array();

return TRUE;

}

return FALSE;

}

/\*\*

\* This callback is executed when a session sets maxlifetime

\* @param int $maxlifetime

\* @param bool $overwrite overwrite previous maxlifetime

\* @return boolean

\*/

public function sessionMaxlifetime($maxlifetime, $overwrite = FALSE)

{

$oCore\_QueryBuilder = Core\_QueryBuilder::update('sessions')

->set('maxlifetime', $maxlifetime)

->where('id', '=', session\_id());

!$overwrite

&& $oCore\_QueryBuilder->where('maxlifetime', '<', $maxlifetime);

$oCore\_QueryBuilder->execute();

// Set cookie with expiration date

//self::\_setCookie();

return TRUE;

}

/\*\*

\* The garbage collector callback is invoked internally by PHP periodically in order to purge old session data.

\* @param string $maxlifetime max life time

\* @return boolean

\*/

public function sessionGc($maxlifetime)

{

Core\_QueryBuilder::delete('sessions')

->where('time + maxlifetime', '<', time())

->execute();

return TRUE;

}

/\*\*

\* Show error

\* @param string $content

\*/

protected function \_error($content)

{

if (Core\_Array::getRequest('\_', FALSE))

{

Core::showJson(array('error' => Core\_Message::get($content, 'error'), 'form\_html' => NULL));

}

else

{

// Service Unavailable

Core\_Response::sendHttpStatusCode(503);

throw new Core\_Exception($content);

}

}

/\*\*

\* Get LOCK name

\* @param int $id session ID

\* @return string

\*/

protected function \_getLockName($id)

{

return function\_exists('hash')

? hash('sha256', $this->\_lockPrefix . '\_' . $id)

: $this->\_lockPrefix . '\_' . $id;

}

/\*\*

\* Lock session

\* @param int $id session ID

\* @return boolean

\*/

protected function \_lock($id)

{

$iStartTime = time();

while (!connection\_aborted())

{

$oDataBase = $this->\_dataBase->setQueryType(0)

->query('SELECT GET\_LOCK(' . $this->\_dataBase->quote($this->\_getLockName($id)) . ', '

. intval($this->\_getLockTimeout) . ') AS `lock`');

$row = $oDataBase->asAssoc()->current();

if (!is\_array($row))

{

$this->\_error('HostCMS session lock error: Get row failure.');

}

if (isset($row['lock']) && $row['lock'] == 1)

{

return TRUE;

}

$iTime = time() - $iStartTime;

if ($iTime > $this->\_lockTimeout)

{

$this->\_error('HostCMS session lock error: Timeout. Please wait! Refreshing page ... <script>setTimeout(function() {window.location.reload(true);}, 1000);</script>');

}

usleep($this->\_nextStepDelay);

}

return FALSE;

}

/\*\*

\* Unlock session

\* @param int $id session ID

\* @return boolean

\*/

protected function \_unlock($id)

{

$oDataBase = $this->\_dataBase->setQueryType(0)

->query('SELECT RELEASE\_LOCK(' . $this->\_dataBase->quote($this->\_getLockName($id)) . ') AS `lock`');

$row = $oDataBase->asAssoc()->current();

if (!is\_array($row))

{

$this->\_error('HostCMS session unlock error: Get row failure');

}

return TRUE;

}

}