Частное учреждение

профессиональная образовательная организация

ТЕХНИКУМ «БИЗНЕС И ПРАВО»

ЗАЩИЩЕНО С ОЦЕНКОЙ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и ФИО руководителя)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

**КУРСОВАЯ РАБОТА**

|  |  |
| --- | --- |
| Тема | Учет медицинских карт |
| По МДК 09.01 | Проектирование и разработка веб-приложений |
| Выполнил | Корнев В.В. |
| Курс, группа | 4 курс, группа ИС-4 |
| Специальность | 09.02.07 Информационные системы и программирование |
| Научный руководитель | Мартыненко В.А. |

г. Белореченск

2022 г.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Введение | 3 |
| 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ | 5 |
| 1.1 Анализ предметной области | 5 |
| 1.2 Цели и задачи выполняемой системы | 5 |
| 1.3 Метод реализации процесса проектирования | 6 |
| 1.4 Требования к системе | 6 |
| 1.5 Обоснование проектных решений | 8 |
| 2. ПРОЕКТНАЯ ЧАСТЬ | 21 |
| 2.1 Информационное обеспечение информационной системы | 21 |
| 2.2 Программное обеспечение информационной системы | 22 |
| 2.3 Технологическое обеспечение информационной системы | 23 |
| ЗАКЛЮЧЕНИЕ | 24 |
| СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ | 25 |
| ПРИЛОЖЕНИЯ |  |

ВВЕДЕНИЕ

Актуальность исследования – успешная работа компании напрямую связанна с деятельностью логистической системой. В зависимости от того, как реализована работа, деятельности основного предприятия, но и от точности работы медицинских карт. Оценка работы. Грамотно структурированная работа – залог успешного развития компании. Актуальность темы заключается в том, что ведение мед учета позволяет сделать работу «прозрачной» и минимизировать временные затраты на проведение различных складских операций.

Объект исследование – информационная система складного учёта.

Предмет исследования – предметом исследования являются процессы размещения, поиска и списания товаров на складе.

Цель исследование – рассмотреть и установить повышение эффективности процессов, добавления, редактирования, удаления, мед карт на на сайте за счет разработки и внедрения информационной учета медицинских карт, с использованием адресного хранения мед карт.

Задачи исследования:

– изучить предметную область;

– рассмотреть ряд существующих разработок для решения поставленной цели;

– выбрать метод реализации работы учета медицинских карт;

– спроектировать работу разрабатываемой системы медицинских карт;

– разработать информационную систему;

– протестировать разработанную информационную систему медицинских карт.

Методы исследования – адресное хранение это способ размещения мед карт в профиле, при котором каждому месту хранения присваивается индивидуальный, номер, или адрес.

Учет медицинских карт – подразумевает постоянный количественный учет по мед картам. Без постоянного учета за картами сложно обеспечить их.

Самый простой способ разместить мед карту в профиле. Все новые пациенты с легкостью смогут оперировать.

Адресная система хранения — это методика размещения мед карт в профиле, (Адрес) позволяет оптимизировать работу предприятия и свести к минимуму влияние человеческого фактора.

1. ОБЩИЕ СВЕДЕНИЯ

1.1 Анализ предметной области

Разрабатываемая система предназначена для более удобной и лёгкой работы с мед картами.

Система будет очень удобной, лёгкой в своём использовании. Для пациентов, администрации будет всё понятно и доступно.

Администратор сайта сможет очень легко и быстро вносить данные о пациенте и наблюдать сколько их. Тем самым будет очень легко и просто следить за пациентами. Учет медицинских карт будет не заторможенным, соответственно будет хорошая работа и пациенты скорее всего больше не вернуться.

1.2 Цели и задачи выполняемой системы

Основными целями создания мед карт на сайте:

– удобное добавление мед карт;

– следить сколько осталось товаров на складе;  
– изменение их параметров;

– удаления мед карт;

Задачами являются:

– Достичь поставленных целей;

– Разработать приложение учет медицинских карт;

– Автоматическое обновление данных;

– Оформить отчёт;

– Освоить навыки полученные на учении;

– Связать программу с БД;

– Сделать структурированную программу.

1.3 Метод реализации процесса проектирования

В ходе проведения работ по разработке учета медицинских карт автоматизируются процессы и проверки. Либо просто добавить ее к пациенту.

При внедрении системы будет достигнуто:

– добавление мед карты;

– редактирование мед карты;

– удаление данных.

– правильная и удобная работа учета медицинских карт.

Объекты автоматизации характеризуются:

– обеспечивать максимальную работоспособность;

– быть надежной и технологичной;

– обслуживание должно проводиться минимальным количеством персонала;

1.4 Требования к системе

Система должная предполагать наличие следующих модулей, представленных на рисунке 1.1:

Изображение выглядит как текст, снимок экрана, линия, Прямоугольник

Автоматически созданное описание

Рисунок 1.1 – Модули которые должны быть в программе

Модули разрабатываемой системы представленные на рисунке 1.2:

Изображение выглядит как снимок экрана, Прямоугольник, линия, текст

Автоматически созданное описание

Рисунок 1.2 – Модули разрабатываемой системы

Описание модулей разрабатываемой системы:

– config – вычисление лимита заброса с базой данных MSQL, указывает число записей на странице;

– database – связь с базой данных MSQL, получение данных;

– style.css – CSS стили;

– strprofile – админ панель;

– categories – имя таблицы и категории склада;

– products – имя таблицы и какие товары находятся на складе;

– AddMedcarts – создание мед карты;

– DeleteMedcart – удаление мед карты;

– index – список главная страница;

– footer – bootstrap, удаление и поверка на удаление;

– header – контейнер таблицы мед карт;

– paging – подсчёт всех товаров в базе данных, чтобы подсчитать общее количество страниц, ссылка на последнюю страницу;

– read\_medcart – страница товара (показ выбранного товара);

– update – редактировать мед карту.

1.5 Обоснование проектных решений

На данном этапе необходимо описать примерную архитектуру разрабатываемой системы.

Для того, чтобы сделать программу для учета медицинских карт, мы будем использовать клиент-серверную архитектуру. Чтобы убедиться, что она нам точно подходит рассмотрим ее подробнее. В клиент-серверной архитектуре имеется три звена:

– Представление данных — на стороне клиента.

– Прикладной компонент — на выделенном сервере приложений, здесь происходит вся бизнес-логика.

– Управление ресурсами — сервер БД (базу данных), который и представляет запрашиваемые данные.

Для данного проекта выбрана эта архитектура, так как у нее есть ряд преимуществ перед другими архитектурами:

– Высокую степень гибкости и масштабируемости.

– Высокую безопасность.

– Высокую производительность (т.к. задачи распределены между Серверами).

MySQL – это система управления базами данных. База данных представляет собой структурированную совокупность данных. Эти данные могут быть любыми - от простого списка предстоящих покупок до перечня экспонатов картинной галереи или огромного количества информации в корпоративной сети.

На рисунке 1.3 представлена структура клиент–серверной архитектуры:

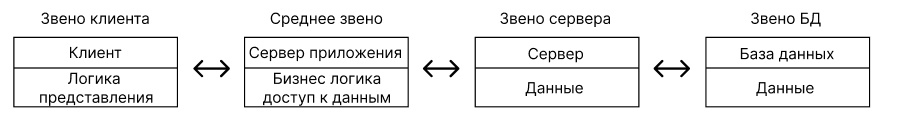


Рисунок 1.3 – Клиент–серверная архитектура

Требования к системе разработаны в соответствии со стандартом качества программного обеспечения ISO 9126:2001, описывающим многоуровневую модель характеристик качества и соответствующий им набор атрибутов.

Требования к системе (товары на складе) представлены в таблице 1.1.

Таблица 1.1 – Требования к системе

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Метки | | Требования |
| 1 | 2 | | 3 |
| Удобство | | | |
| Простота использования | Среднее время, необходимое  пользователю, чтобы найти  необходимый элемент в программе | | Поиск отдельного элемента интерфейса не должен занимать более 1 секунды |
| Привлекательность | Соответствие интерфейса  требованиям | | 99% элементов должны быть работоспособны для  пользователя |
| Обучаемость | Показатель, затрачиваемый  пользователями на обучение  работе | | Пользователь должен изучить работу ПО за первые 3-5 минут  использования |
| Производительность | | | |
| 1 | 2 | 3 | |
| Временная эффективность | Время выполнения компонента программы | - Приложение должно  обслуживать любую функцию не дольше 1 сек  (не включая задержки в сети); | |
| Эффективность использования ресурсов | Объемы ресурсов требуемых для выполнения задач | - Постоянная память,  используемая программой – не более 10МБ;  - Приложение должно потреблять не более 3МБ  памяти на каждый неактивный сеанс с  пользователем; | |
| Переносимость | | | |
| 1 | 2 | 3 | |
| Удобство установки | Легкость установки | Необходима установка:  xampp, mysql, admin php | |
| Способность к сосуществованию | Способность ПО сосуществовать с другими  программами в общем окружении | Программа не должна  занимать большие объемы памяти устройства (максимум 3МБ) | |
| Удобство замены другого ПО данным | Совместимость структур данных | - Система должна быть  разработана по общим  принципам и не требовать навыков от пользователя; - Должна поддерживать  внедрение новых версий;  - Система должна быть применима вместо других  программных систем для решения тех же задач в определенном окружении. | |

Требования к численности и квалификации персонала системы и режиму его работы:

В системе существуют 2 группы пользователей: пациенты и администраторы. Администратор выполняет функцию: просмотра, занесения данных, редактирование, удаления, обновления.

Администратор выполняет функцию: просмотра, занесение данных, обновления, доступ к функциям БД.

Пользователи системы должны иметь опыт работы с персональным компьютером на базе операционных систем Microsoft Windows на уровне пользователя.

В таблице 1.2 представлены требования к надежности.

Таблица 1.2 – Требования к надёжности

|  |  |  |
| --- | --- | --- |
| Атрибуты | Метрики | Требования |
| Надёжность | | |
| Устойчивость к отказу | Коэффициент аварийных  отказов | - Не менее 90% ошибок  в программе должны  обрабатываться без  экстренного завершения;  - Обеспечение бесперебойного питание  активного сетевого  оборудования; |
| Способность к восстановлению | Отношение количества числа успешного восстановление работы системы к ее аварийному завершению за определённый промежуток времени | - Восстановление  работоспособности  системы не должно превышать 5 минут; |
| Уровень зрелости | Среднее время работы без сбоев | - Система должна  работать 24 часа в сутки;  - Система должна оповещать пользователя при некорректных действиях в системе;  - Данные, которые вводит пользователь должны сохраняться. |

Средняя доступность приложения должна составлять не менее 99%

Среднее время между сбоями — это среднее время, за которое компонент или модуль может выполнять свои функции без перерыва.

Измеряется от начала работы до момента следующего сбоя.

Среднее временя работы без сбоев должно составлять не менее 1500 часов.

В таблице 1.3 представлен перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

Таблица 1.3 – Перечень и критерии отказов для каждой функции

|  |  |
| --- | --- |
| Функции | Критерии отказа |
| Отображение данных | Отсутствие соединения с базой данных |
| Просмотр списка товаров по одному | Отсутствие соединения с базой данных |
| Поиск | Отсутствие соединения с базой данных |
| Добавление | Отсутствие соединения с базой данных |
| Редактирование | Отсутствие соединения с базой данных |
| Удаление | Отсутствие соединения с базой данных |

Требования к функциям сайта учета медицинских карт, выполняемым программным обеспечением

Имя: Вход пациента

Описание: Один из специалистов мог зайти по уникальному адресу

Действующий субъект: Админ

Предусловие: специалист Админ зашел на страницу учета мед карт

Сценарий:

– Админу предоставляется персональный компьютер

– пациент заходит на сайт по уникальному адресу

– пациент успешно зашёл на сайт

Расширения: в случае некорректного формата введенных данных в БД, при попытки зайти по уникальному адресу, выведется сообщение с описанием ошибки «Ошибка подключения».

На рисунке 1.4 представлена диаграмма последовательности сценария "Входа сотрудника".

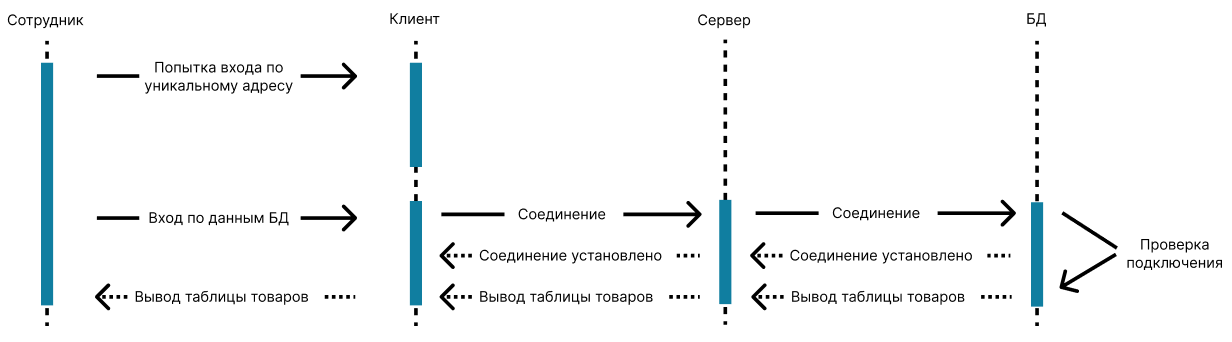


Рисунок 1.4 – Диаграмма последовательности сценария "Входа Администратора"

Имя: поиск мед карт

Описание: админ имеет возможность найти нужный ему мед карту на учете мед карт

Действующий субъект: Админ

Предусловие: Админ успешно зашел в систему, используя уникальный адрес

Сценарий:

– система отображает весь список мед карт

– Админ в поле поиска пишет то, какая мед карта ему нужена и он хочет найти именно ее

– Админ нажимает на кнопку «Поиск»

– система обновляет приложение и выводит нужную мед карт, в названии которых есть, то, что ввел Админ

Расширения: при отсутствии соединения с сервером при нажатии на кнопку «Поиск» выведется сообщение об ошибке.

На рисунке 1.5 представлена диаграмма последовательности сценария "Поиска товара".

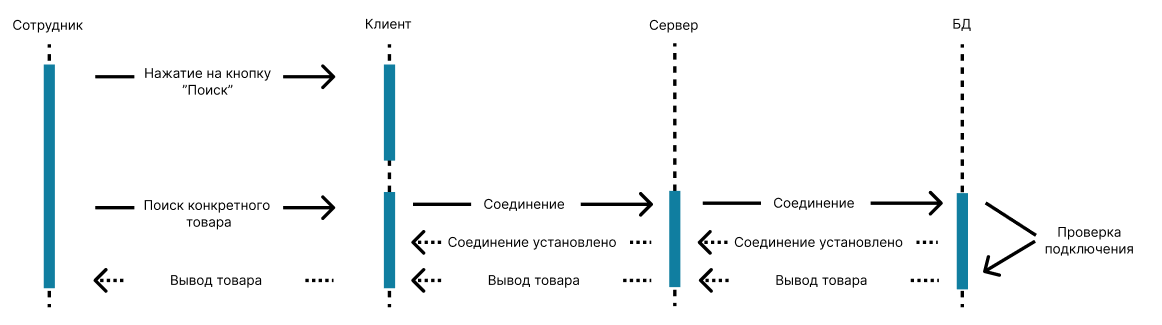


Рисунок 1.5 – Диаграмма последовательности сценария "Поиска мед карт"

Имя: добавление

Описание: Админ имеет возможность добавить новые данные

Действующий субъект: Админ

Предусловие: Админ успешно зашел в систему, используя уникальный адрес

Сценарий:

– система добавляет новые данные

– Админ в полях указывает данные

– Админ нажимает на кнопку «Добавить мед карту»

– система обновляет приложение и выводит добавленные данные. Те, что ввел сотрудник

Расширения: при отсутствии соединения с сервером при нажатии на кнопку «Добавить» выведется сообщение об ошибке.

На рисунке 1.6 представлена диаграмма последовательности сценария

"Добавления".

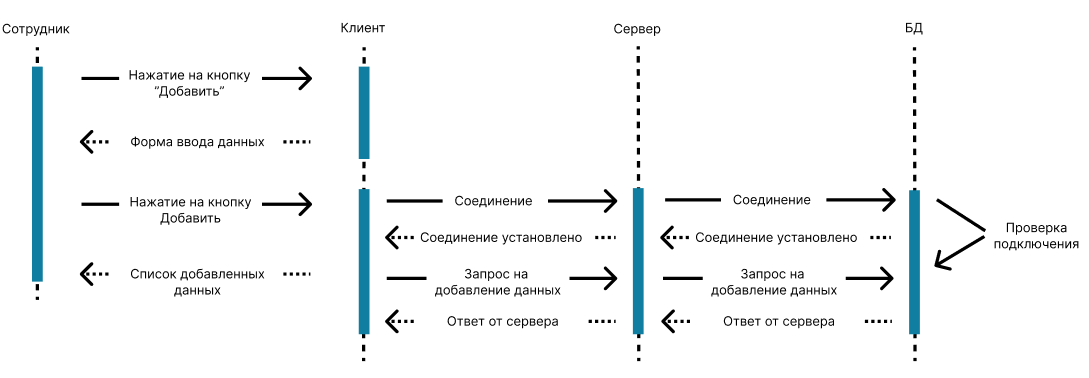


Рисунок 1.6 – Диаграмма последовательности сценария "Добавления"

Имя: редактирование данных

Описание: сотрудник имеет возможность редактировать данные

Действующий субъект: Админ

Предусловие: Админ успешно зашел в систему, используя уникальный адрес

Сценарий:

– система редактирует мед карты

– Админ вносит в поля редактирования ту информацию, на какую хочет заменить

– Админ нажимает на кнопку «Редактировать»

– система обновляет сайт и выводит товар с заменёнными данными

Расширения: при отсутствии соединения с сервером при нажатии на кнопку «Редактировать» выведется сообщение об ошибке.

На рисунке 1.7 представлена диаграмма последовательности сценария "Редактирования".

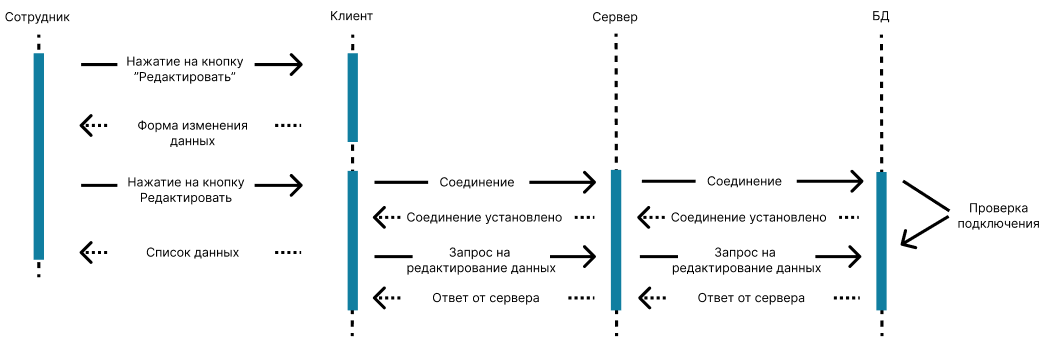


Рисунок 1.7 – Диаграмма последовательности сценария "Редактирования"

Имя: удаление данных

Описание: сотрудник имеет возможность удалить данные

Действующий субъект: Админ

Предусловие: сотрудник успешно зашел в систему, используя уникальный адрес

Сценарий:

– система удаления мед карты

– Админ выбирает товар который хочет удалить

– Админ нажимает на кнопку «Удалить»

– система обновляет данные и удаляет выбранный товар

Расширения: при отсутствии соединения с сервером при нажатии на кнопку «Удаление» выведется сообщение об ошибке

На рисунке 1.7 представлена диаграмма последовательности сценария "Удаления".

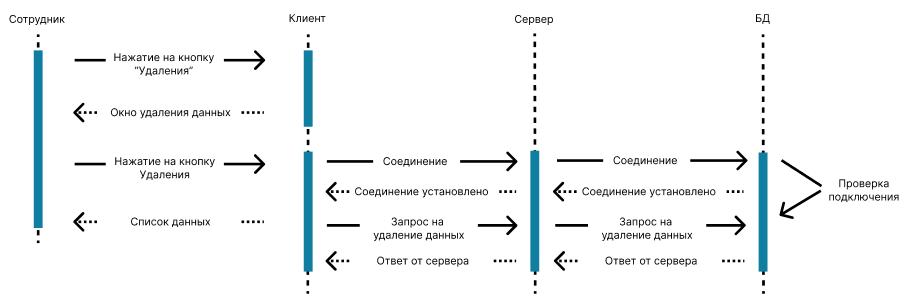


Рисунок 1.7 – Диаграмма последовательности сценария "Удаления"

Для функционирования системы требуется наличие сервера и клиентских компьютеров, на которых будет производится работа с системой. Перед вводом системы в действие необходимо:

– интернет соединение;

– обеспечить каждого специалиста персональным компьютером;

– установить на каждый ПК данную программу;

– иметь программу xampp;

– база данных MySQL.

При анализе предметной области разрабатываемой системы были выделены следующие сущности, на основе которых будет разрабатываться база данных системы:

– Медицинские карты;

– Данные к каждой мед карте пациента;

– Описание данных.

Разрабатываемая система требует в своей архитектуре наличие БД-сервера со свободной реляционной системой управления – MySQL.

База данных информационной системы должна состоять из следующих связанных таблиц:

– categories – таблица содержащая информацию категории товаров.

– products – таблица содержащая информацию о товарах на складе.

Данные о категории товаров представлены в таблице categories 1.4.

Таблица 1.4 – Данные о медицинских картах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | int | 11 | Идентификатор категории | Первичный ключ |
| name | varchar | 256 | Название категории | NOT NULL |
| created | datetime | - | Когда был создан товар | NOT NULL |
| modified | timestamp | - | Когда был изменён товар | NOT NULL |

Данные о товаре должны размещаться в таблице products 1.5.

Таблица 1.5 – Данные о товаре на складе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | int | 11 | Идентификатор категории | Первичный ключ |
| name | varchar | 256 | Название товара | NOT NULL |
| description | text | - | Количество товара на складе | NOT NULL |
| price | varchar | 256 | Цена товара | NOT NULL |
| category\_id | int | 11 | Id категории, к чему относится | NOT NULL |
| created | datetime | - | Когда был создан товар | NOT NULL |
| modified | timestamp | - | Когда был изменён товар | NOT NULL |

На рисунке 1.8 представлено окно добавления товара на склад.

В данном окне указывается название товара, его стоимость, количество на складе, выбор категории (именно к какой категории относится данный товара для корректного занесения в БД).

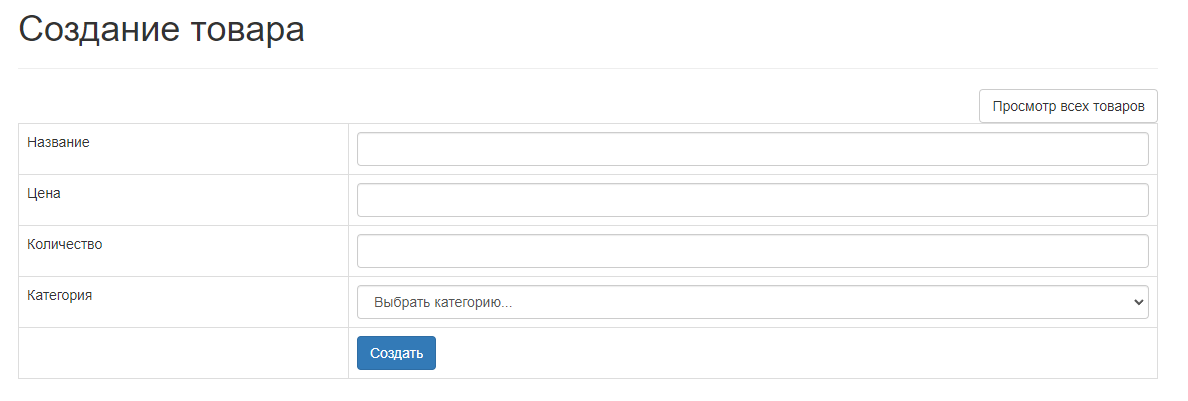


Рисунок 1.8 – Окно добавление товара на склад

На рисунке 1.9 представлено окно просмотра всех товаров на складе.

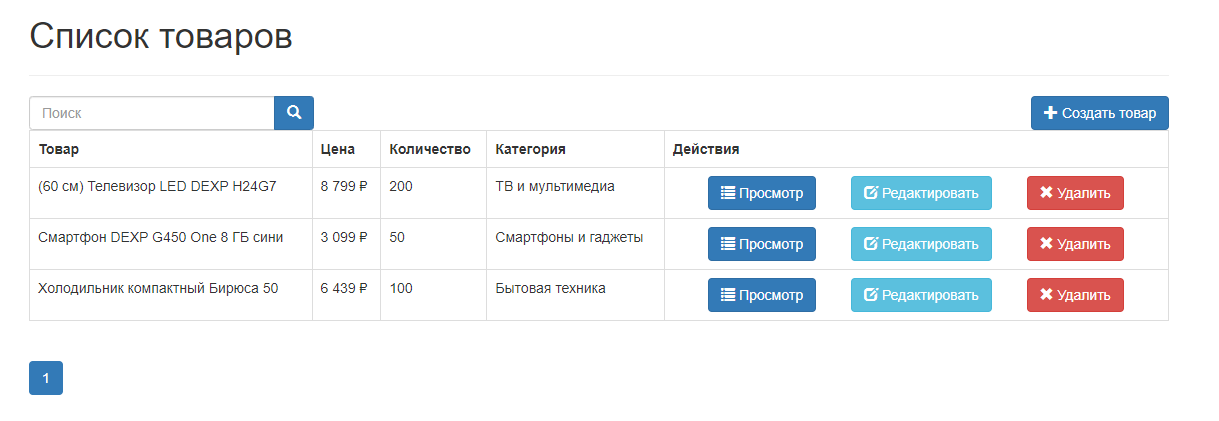


Рисунок 1.9 – Список товаров на складе

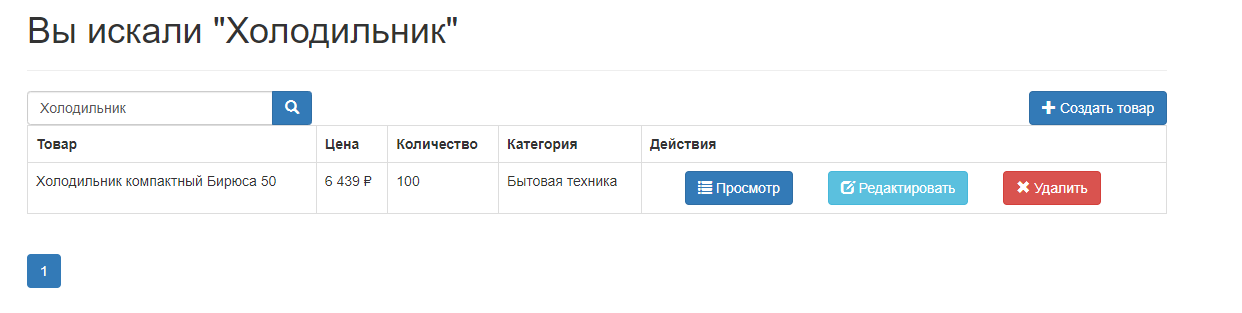
На рисунке 2.1 представлено окно поиска товара

Рисунок 2.1 – Поиск товара на складе

На рисунке 2.2 – представлено окно просмотра выбранного товара.

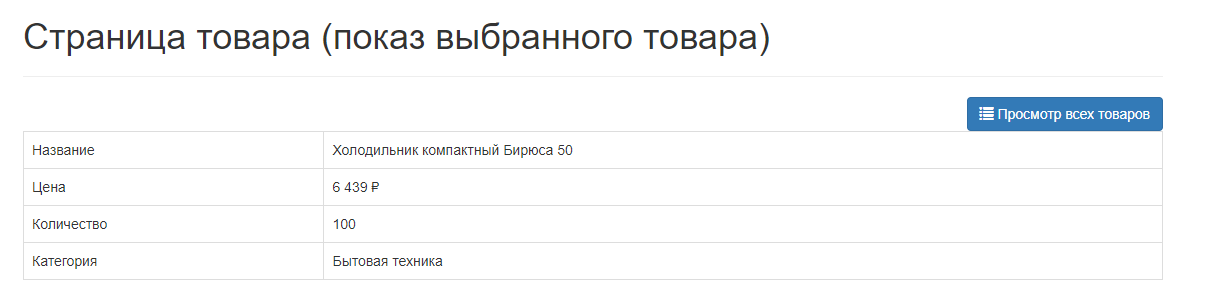


Рисунок 2.3 – Окно показа выбранного товара

На рисунке 2.3 – представлено окно удаления товара.

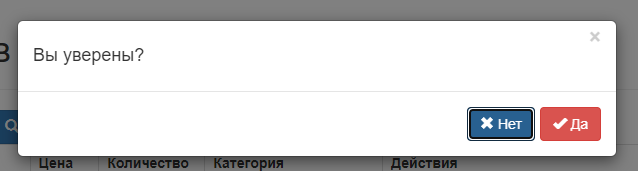


Рисунок 2.3 – Окно удаления товара

На рисунке 2.4 – представлено окно редактирования товара.

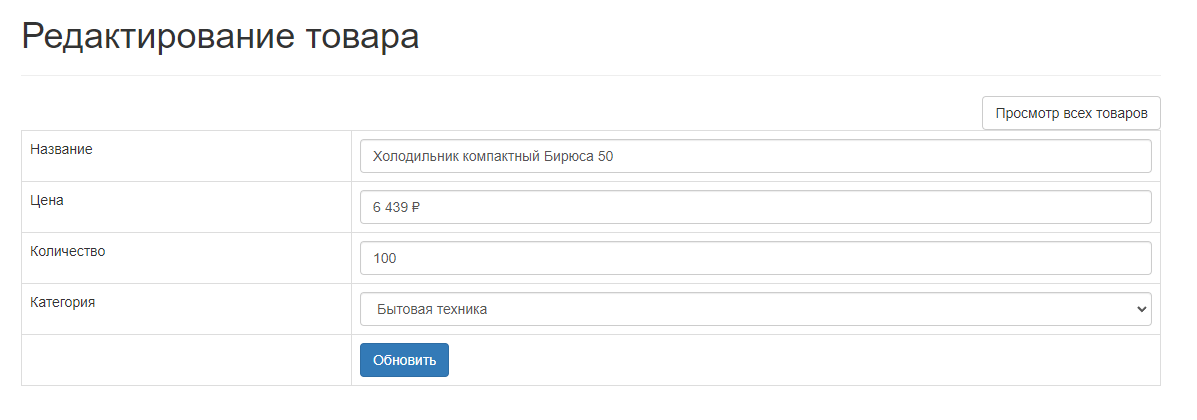


Рисунок 2.4 – Окно редактирования товара

На данном этапе разберем, то как будем вести себя система при выполнение выделенных модулей. Для этого мы будем использовать диаграммы состояний в нотации UML.

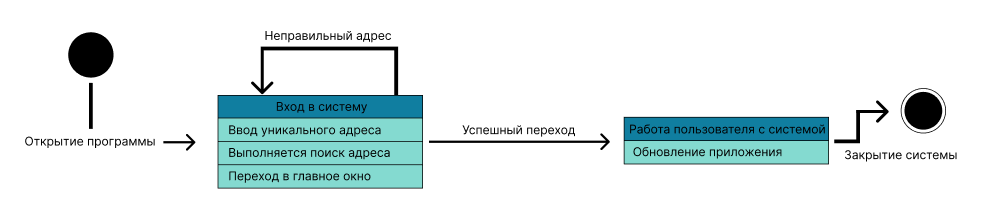
На рисунке 2.5 представлена общая диаграмма состояний. При переходе пользователем в программу пользователь будет находиться в том, состоянии, когда ему необходимо войти, чтобы продолжить работу с системой и осуществлять деятельность согласно выделенным вариантам использования.

Рисунок 2.5 – Общая диаграмма состояний системы

На рисунке 2.6 показано детальное описание работы пользователя с системой. Сотрудник может просматривать список всех данных, добавлять, удалять, редактировать, производить поиск.

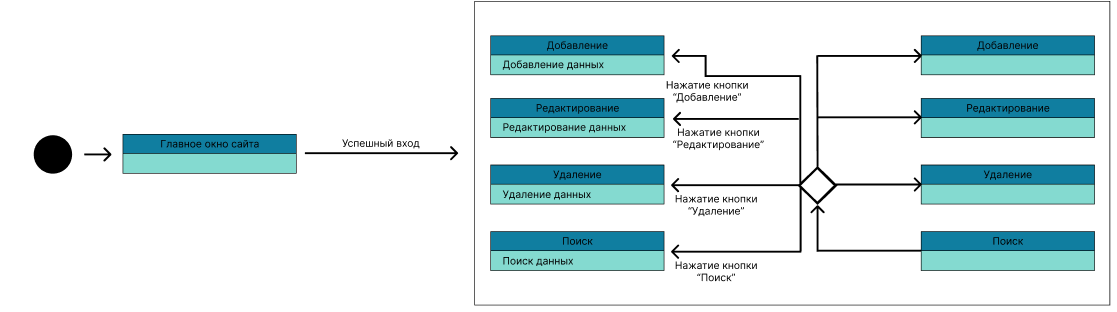


Рисунок 2.6 – Диаграмма состояний работы пользователя с системой

На рисунке 2.7 изображена декомпозиция состояния «Добавление». После того как сотрудник успешно зашел в программу, ему необходимо выбрать добавление, которое ему необходимо. Далее он может заполнить поля данными, нажав на Кнопку «Добавить» добавятся данные в базу данных.

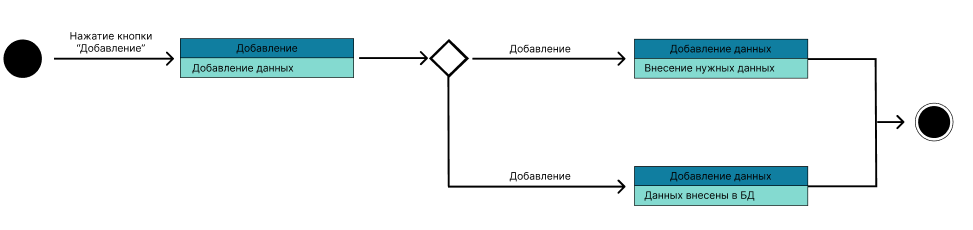


Рисунок 2.7 – Диаграмма состояний добавление данных

На рисунке 2.8 изображена декомпозиция состояния «Редактирования».

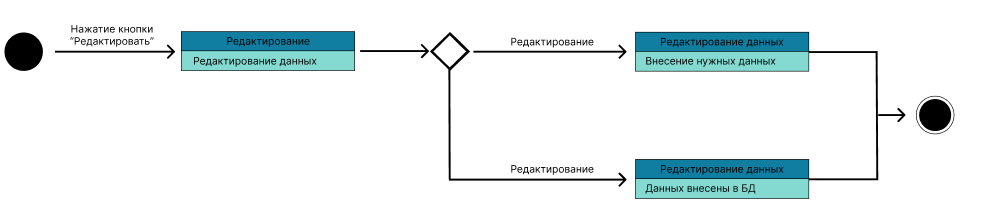
Сотрудник должен нажать на кнопку «Редактировать». После этого приложение обновится и обновит данные в базе данных. Далее сотрудник может просмотреть все изменённые данные.

Рисунок 2.8 – Диаграмма состояний редактирования данных

На рисунке 2.9 изображена диаграмма состояния «Удаления». Сотрудник может удалять все данные товара. Далее необходимо нажать на кнопку «Удалить» и программа выполнит запрос.

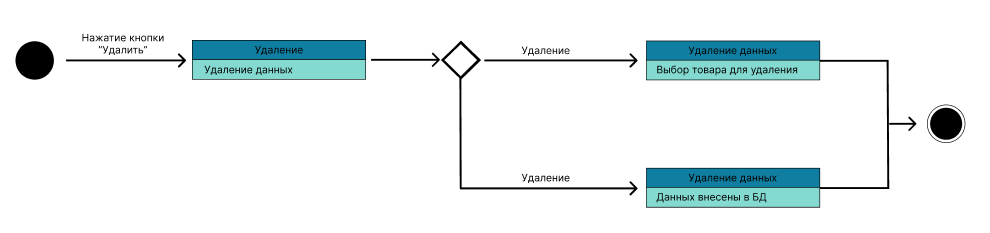


Рисунок 2.9 – Диаграмма состояний удаления данных

На рисунке 3.1 изображена диаграмма состояния «Поиска». Сотрудник может делать поиск по всем типам данных. Далее необходимо нажать на кнопку «Поиск» и программа выполнит запрос поиск.

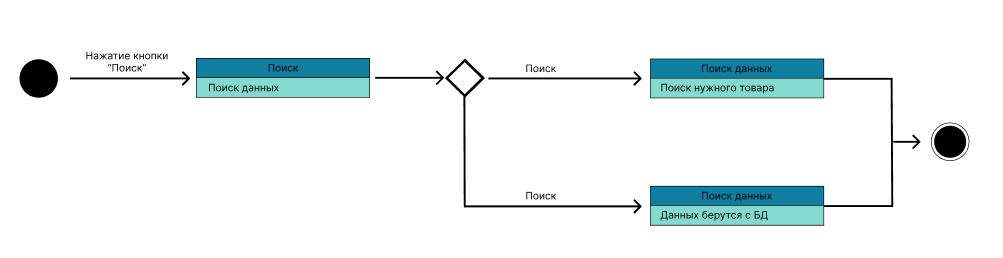


Рисунок 3.1 – Диаграмма состояний поиска данных

2. ПРОЕКТНАЯ ЧАСТЬ

2.1 Информационное обеспечение информационной системы

Все данные сайта должны храниться в структурированном виде под управлением реляционной СУБД. Исключения составляют файлы данных, предназначенные для просмотра и скачивания (изображения, видео, документы и т.п.). Такие файлы сохраняются в файловой системе, а в БД размещаются ссылки на них.

Наполнение различных сайтов, функционирование которых поддерживается одной и той же инсталляцией системы, должно храниться под управлением единой СУБД.

Система управления базами данных (СУБД) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными. Иными словами, СУБД — это набор программ, позволяющий организовывать, контролировать и администрировать базы данных. Большинство сайтов не могут функционировать без базы данных, поэтому СУБД используется практически повсеместно.

Основные функции СУБД:

– управление данными во внешней памяти (на дисках);

– управление данными в оперативной памяти с использованием дискового кэша;

– журнализация изменений (сохранение истории), резервное копирование и восстановление базы данных после сбоев;

– поддержка языков БД (язык определения данных, язык манипулирования данными).

Для реализации статических страниц и шаблонов должны использоваться языки HTML 4.0 и CSS.

HTML (от английского HyperText Markup Language) — это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.

HTML состоит из тегов — команд, которые указывают браузеру, как отображать помещённый в них текст. Это и есть элементы веб-страницы. У каждого тега есть имя, которое заключается в угловые скобки < и >.

Для реализации интерактивных элементов сервероной части должны использоваться языки php, javascript.

Для реализации динамических страниц должен использоваться язык PHP.

Объем одной стандартной загружаемой страницы сайта в среднем не должен превышать 170 kb.

Объем flash-заставки не должен превышать 300 Kb.

CSS (Cascading Style Sheets, каскадные таблицы стилей) — язык описания внешнего вида HTML-документа. Это одна из базовых технологий в современном интернете. Практически ни один сайт не обходится без CSS, поэтому HTML и CSS действуют в единой связке.

Каскадные таблицы стилей работают с HTML, но это совершенно другой язык. HTML структурирует документ и упорядочивает информацию, а CSS взаимодействует с браузером, чтобы придать документу оформление.

2.2 Программное обеспечение информационной системы

Для функционирования сайта необходимо следующее программное обеспечение:

– Операционная система: Windows 10/Linux/macOS;

– Сервер: MS SQL Server 2019;

– СУБД: MSSQL для MS SQL Server;

– XAMPP.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 2022, которая вышла в ноябре 2022 году и которая будет использоваться в текущем руководстве.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

Необходимое программное обеспечение для персонального компьютера:

- Операционная система: Windows 10/Linux/macOS

Сайт должен быть доступен для полнофункционального просмотра с помощью следующих браузеров:

– Opera 6.0 и выше;

– Google;

– Yandex.

JavaScript - предназначен для написания сценариев для активных HTML-страниц. Язык JavaScript не имеет никакого отношения к языку Java. Java разработан фирмой SUN. JavaScript - фирмой Netscape Communication Corporation. Первоначальное название - LiveScript. После завоевания языком Java всемирной известности LiveScript из коммерческих соображений переименовали в JavaScript.

2.3 Технологическое обеспечение информационной системы

Для функционирования сайта необходимо следующее техническое обеспечение со следующими минимальными характеристиками:

– процессор – intel core i3;

– оперативная память – 512 Mb RAM;

ЗАКЛЮЧЕНИЕ

При разработке приложения было сделано простота использования и удобство интерфейса, а также защищенность от постороннего вмешательства.

Также были достигнуты цели курсовой работы. Удалось создать трёхуровневую архитектуру приложения. Цель работы достигнута, а именно создано приложение для работы с базой данных.

Куда проще вносить данные о товаре и хранить их там.

Задачи решены, а именно создать трёхуровневую архитектуру приложения, быстрый доступ к базе данных, добавлению товара, просмотра, редактирования и удаления любого товара.

В результате выполнения курсовой, был создан динамический сайт для внесения и хранения там данных о товарах. Для его разработки применялось актуальное программное обеспечение, без помощи которого не было бы возможности добиться поставленной цели. Это программное обеспечение включает в себя:

– язык программирования PHP;

– фреймворк bootstrap;

– веб-сервер XAMPP;

– реляционная система управления БД MySQL;

– приложение PHPMyAdmin.

С помощью этих инструментов значительно сокращена ручная работа и повторяющийся код. Чтобы добиться ЧПУ без применения bootstrap и дополнительных библиотек, необходимо вносить много изменений в файлы.

Таким образом, затрачено меньше времени на разработку сайта. Всё информационное наполнение сайта находится в БД.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вандюк, Джон К. CMS Drupal. Руководство по разработке системы управления сайтом / Вандюк, Джон К., Мэтт Вестгейт,. - М.: Вильямс, 2019. - 400 c.
2. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Триумф, 2020. - 464 c.
3. Дакетт, Джон HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. - Москва: Наука, 2019. - 480 c.
4. Дронов, В. JavaScript в Web-дизайне / В. Дронов. - М.: СПб: БХВ, 2019. - 880 c.
5. Дронов, В. PHP, MySQL и Dreamweaver. Разработка интерактивных Web-сайтов / В. Дронов. - М.: БХВ-Петербург, 2019. - 480 c.
6. Дронов, В.А. PHP 5/6, MySQL 5/6 и Dreamweaver CS4. Разработка интерактивных Web-сайтов / В.А. Дронов. - М.: БХВ-Петербург, 2019. - 820 c.
7. Дронов, Владимир JavaScript и AJAX в Web-дизайне / Владимир Дронов. - Москва: Высшая школа, 2019. - 736 c.
8. Дронов, Владимир PHP, MySQL и Dreamweaver MX 2004. Разработка интерактивных Web-сайтов / Владимир Дронов. - М.: "БХВ-Петербург", 2019. - 448 c.
9. Колисниченко, Д. PHP 5/6 и MySQL 6. Разработка Web-приложений / Д. Колисниченко. - М.: БХВ-Петербург, 2021. - 560 c.
10. Колисниченко, Д.Н. PHP 5/6 и MySQL 6. Разработка Web-приложений / Д.Н. Колисниченко. - М.: БХВ-Петербург, 2021. - 364 c.
11. Кузнецов PHP. Практика создания Web-сайтов / Кузнецов, М.В. и. - М.: БХВ-Петербург, 2020. - 895 c.
12. Кузнецов, М. PHP 5. Практика создания Web-сайтов / М. Кузнецов, И. Симдянов, С. Голышев. - М.: БХВ-Петербург, 2020. - 960 c.
13. Леонтьев, Борис PHP 5.0 для начинающих, или как создать динамический WEB-сайт / Борис Леонтьев. - М.: Новый издательский дом, 2019. - 176 c.
14. Петюшкин, Алексей HTML в Web-дизайне / Алексей Петюшкин. - Москва: Машиностроение, 20. - 400 c.
15. Шкрыль, А. PHP - это просто. Программируем для Web-сайта / А. Шкрыль. - М.: БХВ-Петербург, 2021. - 368 c.

ПРИЛОЖЕНИЯ

CSS

.left-margin {

margin: 0 .5em 0 0;

}

.right-button-margin {

margin: 0 0 1em 0;

overflow: hidden;

}

/\* modal \*/

.modal-body {

padding: 20px 20px 0px 20px !important;

text-align: center !important;

}

.modal-footer {

text-align: center !important;

}

Core

<?php

// страница, указанная в параметре URL, страница по умолчанию: 1

$page = isset($\_GET["page"]) ? $\_GET["page"] : 1;

// укажем число записей на странице

$records\_per\_page = 5;

// вычисление лимита запроса

$from\_record\_num = ($records\_per\_page \* $page) - $records\_per\_page;

Database

<?php

class Database

{

// укажите свои собственные учетные данные для базы данных

private $host = "localhost";

private $db\_name = "php-oop-mysql";

private $username = "test123";

private $password = "777";

public $conn;

// получение соединения с базой данных

public function getConnection()

{

$this->conn = null;

try {

$this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db\_name, $this->username, $this->password);

} catch (PDOException $exception) {

echo "Ошибка соединения: " . $exception->getMessage();

}

return $this->conn;

}

}

Category

<?php

class Category

{

// подключение к базе данных и имя таблицы

private $conn;

private $table\_name = "categories";

// свойства объекта

public $id;

public $name;

public function \_\_construct($db)

{

$this->conn = $db;

}

// данный метод используется в раскрывающемся списке

function read()

{

// запрос MySQL: выбираем столбцы в таблице «categories»

$query = "SELECT

id, name

FROM

" . $this->table\_name . "

ORDER BY

name";

$stmt = $this->conn->prepare($query);

$stmt->execute();

return $stmt;

}

// получение названия категории по её ID

function readName()

{

// запрос MySQL

$query = "SELECT name FROM " . $this->table\_name . " WHERE id = ? limit 0,1";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(1, $this->id);

$stmt->execute();

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

$this->name = $row["name"];

}

}

Product

<?php

class Product

{

// подключение к базе данных и имя таблицы

private $conn;

private $table\_name = "products";

// свойства объекта

public $id;

public $name;

public $price;

public $description;

public $category\_id;

public $timestamp;

public function \_\_construct($db)

{

$this->conn = $db;

}

// метод создания товара

function create()

{

// запрос MySQL для вставки записей в таблицу БД «products»

$query = "INSERT INTO

" . $this->table\_name . "

SET

name=:name, price=:price, description=:description, category\_id=:category\_id, created=:created";

$stmt = $this->conn->prepare($query);

// опубликованные значения

$this->name = htmlspecialchars(strip\_tags($this->name));

$this->price = htmlspecialchars(strip\_tags($this->price));

$this->description = htmlspecialchars(strip\_tags($this->description));

$this->category\_id = htmlspecialchars(strip\_tags($this->category\_id));

// получаем время создания записи

$this->timestamp = date("Y-m-d H:i:s");

// привязываем значения

$stmt->bindParam(":name", $this->name);

$stmt->bindParam(":price", $this->price);

$stmt->bindParam(":description", $this->description);

$stmt->bindParam(":category\_id", $this->category\_id);

$stmt->bindParam(":created", $this->timestamp);

if ($stmt->execute()) {

return true;

} else {

return false;

}

}

// метод для получения товаров

function readAll($from\_record\_num, $records\_per\_page)

{

// запрос MySQL

$query = "SELECT

id, name, description, price, category\_id

FROM

" . $this->table\_name . "

ORDER BY

name ASC

LIMIT

{$from\_record\_num}, {$records\_per\_page}";

$stmt = $this->conn->prepare($query);

$stmt->execute();

return $stmt;

}

// используется для пагинации товаров

public function countAll()

{

// запрос MySQL

$query = "SELECT id FROM " . $this->table\_name . "";

$stmt = $this->conn->prepare($query);

$stmt->execute();

$num = $stmt->rowCount();

return $num;

}

// метод для получения товара

function readOne()

{

// запрос MySQL

$query = "SELECT

name, price, description, category\_id

FROM

" . $this->table\_name . "

WHERE

id = ?

LIMIT

0,1";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(1, $this->id);

$stmt->execute();

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

$this->name = $row["name"];

$this->price = $row["price"];

$this->description = $row["description"];

$this->category\_id = $row["category\_id"];

}

// метод для обновления товара

function update()

{

// MySQL запрос для обновления записи (товара)

$query = "UPDATE

" . $this->table\_name . "

SET

name = :name,

price = :price,

description = :description,

category\_id = :category\_id

WHERE

id = :id";

// подготовка запроса

$stmt = $this->conn->prepare($query);

// очистка

$this->name = htmlspecialchars(strip\_tags($this->name));

$this->price = htmlspecialchars(strip\_tags($this->price));

$this->description = htmlspecialchars(strip\_tags($this->description));

$this->category\_id = htmlspecialchars(strip\_tags($this->category\_id));

$this->id = htmlspecialchars(strip\_tags($this->id));

// привязка значений

$stmt->bindParam(":name", $this->name);

$stmt->bindParam(":price", $this->price);

$stmt->bindParam(":description", $this->description);

$stmt->bindParam(":category\_id", $this->category\_id);

$stmt->bindParam(":id", $this->id);

// выполняем запрос

if ($stmt->execute()) {

return true;

}

return false;

}

// метод для удаления товара

function delete()

{

// запрос MySQL для удаления

$query = "DELETE FROM " . $this->table\_name . " WHERE id = ?";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(1, $this->id);

if ($result = $stmt->execute()) {

return true;

} else {

return false;

}

}

// выбираем товары по поисковому запросу

public function search($search\_term, $from\_record\_num, $records\_per\_page)

{

// запрос к БД

$query = "SELECT

c.name as category\_name, p.id, p.name, p.description, p.price, p.category\_id, p.created

FROM

" . $this->table\_name . " p

LEFT JOIN

categories c

ON p.category\_id = c.id

WHERE

p.name LIKE ? OR p.description LIKE ?

ORDER BY

p.name ASC

LIMIT

?, ?";

// подготавливаем запрос

$stmt = $this->conn->prepare($query);

// привязываем значения переменных

$search\_term = "%{$search\_term}%";

$stmt->bindParam(1, $search\_term);

$stmt->bindParam(2, $search\_term);

$stmt->bindParam(3, $from\_record\_num, PDO::PARAM\_INT);

$stmt->bindParam(4, $records\_per\_page, PDO::PARAM\_INT);

// выполняем запрос

$stmt->execute();

// возвращаем значения из БД

return $stmt;

}

// метод для подсчёта общего количества строк

public function countAll\_BySearch($search\_term)

{

// запрос

$query = "SELECT

COUNT(\*) as total\_rows

FROM

" . $this->table\_name . " p

WHERE

p.name LIKE ? OR p.description LIKE ?";

// подготовка запроса

$stmt = $this->conn->prepare($query);

// привязка значений

$search\_term = "%{$search\_term}%";

$stmt->bindParam(1, $search\_term);

$stmt->bindParam(2, $search\_term);

$stmt->execute();

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

return $row["total\_rows"];

}

}

create\_product

<?php

// подключим файлы, необходимые для подключения к базе данных и файлы с объектами

include\_once "config/database.php";

include\_once "objects/product.php";

include\_once "objects/category.php";

// получаем соединение с базой данных

$database = new Database();

$db = $database->getConnection();

// создадим экземпляры классов Product и Category

$product = new Product($db);

$category = new Category($db);

$page\_title = "Создание товара";

require\_once "layout\_header.php";

?>

<div class="right-button-margin">

<a href="index.php" class="btn btn-default pull-right">Просмотр всех товаров</a>

</div>

<?php

// если форма была отправлена

if ($\_POST)

{

// установим значения свойствам товара

$product->name = $\_POST["name"];

$product->price = $\_POST["price"];

$product->description = $\_POST["description"];

$product->category\_id = $\_POST["category\_id"];

// создание товара

if ($product->create()) {

echo '<div class="alert alert-success" style="margin-top: 70px">Товар был успешно создан.</div>';

}

// если не удается создать товар, сообщим об этом пользователю

else {

echo '<div class="alert alert-danger" style="margin-top: 70px">Невозможно создать товар.</div>';

}

}

?>

<!-- HTML-формы для создания товара -->

<form action="<?= htmlspecialchars($\_SERVER["PHP\_SELF"]) ?>" method="post">

<table class="table table-hover table-responsive table-bordered">

<tr>

<td>Название</td>

<td><input type="text" name="name" class="form-control" /></td>

</tr>

<tr>

<td>Цена</td>

<td><input type="text" name="price" class="form-control" /></td>

</tr>

<tr>

<td>Количество</td>

<td><input type="text" name="description" class="form-control" /></td>

</tr>

<tr>

<td>Категория</td>

<td>

<?php

// читаем категории товаров из базы данных

$stmt = $category->read();

// помещаем их в выпадающий список

echo "<select class='form-control' name='category\_id'>";

echo "<option>Выбрать категорию...</option>";

while ($row\_category = $stmt->fetch(PDO::FETCH\_ASSOC)) {

extract($row\_category);

echo "<option value='{$id}'>{$name}</option>";

}

echo "</select>";

?>

</td>

</tr>

<tr>

<td></td>

<td>

<button type="submit" class="btn btn-primary">Создать</button>

</td>

</tr>

</table>

</form>

<?php // подвал

require\_once "layout\_footer.php";

?>

delete\_product

<?php

// проверим, было ли получено значение в $\_POST

if ($\_POST) {

// подключаем файлы для работы с базой данных и файлы с объектами

include\_once "config/database.php";

include\_once "objects/product.php";

// получаем соединение с базой данных

$database = new Database();

$db = $database->getConnection();

// подготавливаем объект Product

$product = new Product($db);

// устанавливаем ID товара для удаления

$product->id = $\_POST["object\_id"];

// удаляем товар

if ($product->delete()) {

echo "Товар был удалён";

}

// если невозможно удалить товар

else {

echo "Невозможно удалить товар";

}

}

Index

<?php

// содержит переменные пагинации

include\_once "config/core.php";

// файлы для работы с БД и файлы с объектами

include\_once "config/database.php";

include\_once "objects/product.php";

include\_once "objects/category.php";

// получение соединения с БД

$database = new Database();

$db = $database->getConnection();

$product = new Product($db);

$category = new Category($db);

$page\_title = "Список товаров";

include\_once "layout\_header.php";

// получение товаров

$stmt = $product->readAll($from\_record\_num, $records\_per\_page);

// укажем страницу, на которой используется пагинация

$page\_url = "index.php?";

// подсчёт общего количества строк (используется для разбивки на страницы)

$total\_rows = $product->countAll();

// контролирует, как будет отображаться список продуктов

include\_once "read\_template.php";

// содержит наш JavaScript и закрывающие теги html

include\_once "layout\_footer.php";

layout\_footer

</div>

<!-- /container -->

<!-- jQuery (необходим для Bootstrap JavaScript) -->

<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<!-- bootstrap JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<!-- bootbox JavaScript -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootbox.js/4.4.0/bootbox.min.js"></script>

<script>

// JavaScript для удаления товара

$(document).on("click", ".delete-object", function() {

const id = $(this).attr("delete-id");

bootbox.confirm({

message: "<h4>Вы уверены?</h4>",

buttons: {

confirm:

label: "<span class='glyphicon glyphicon-ok'></span> Да",

className: "btn-danger"

},

cancel: {

label: "<span class='glyphicon glyphicon-remove'></span> Нет",

className: "btn-primary"

}

},

callback: function(result) {

if (result == true) {

$.post("delete\_product.php", {

object\_id: id

}, function(data) {

location.reload();

}).fail(function() {

alert("Невозможно удалить.");

});

}

}

});

return false;

});

</script>

</body>

</html>

layout\_header

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title><?= $page\_title ?></title>

<!-- bootstrap CSS -->

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

<!-- кастомный CSS -->

<link rel="stylesheet" href="libs/css/custom.css" />

</head>

<body>

<!-- container -->

<div class="container">

<!-- page header -->

<div class="page-header">

<h1><?= $page\_title ?></h1>

</div>

Paging

<?php

echo "<ul class='pagination'>";

// button for first page

if ($page > 1) {

echo "<li><a href='{$page\_url}' title='Перейти к первой странице'>Первая</a></li>";

}

// подсчёт всех товаров в БД, чтобы подсчитать общее количество страниц

$total\_pages = ceil($total\_rows / $records\_per\_page);

// диапазон ссылок для показа

$range = 2;

// отображаем ссылки на "диапазон страниц" вокруг "текущей страницы"

$initial\_num = $page - $range;

$condition\_limit\_num = ($page + $range) + 1;

for ($x = $initial\_num; $x < $condition\_limit\_num; $x++) {

// убедимся, что "$x больше 0" И "меньше или равно $total\_pages"

if (($x > 0) && ($x <= $total\_pages)) {

// текущая страница

if ($x == $page) {

echo "<li class='active'><a href='#'>$x <span class='sr-only'>(current)</span></a></li>";

}

// НЕ текущая страница

else {

echo "<li><a href='{$page\_url}page=$x'>$x</a></li>";

}

}

}

// ссылка по последнюю страницу

if ($page < $total\_pages) {

echo "<li><a href='{$page\_url}page={$total\_pages}' title='Перейти к последней странице из {$total\_pages}'>";

echo "Последняя";

echo "</a></li>";

}

echo "</ul>";

read\_product

<?php

// установка заголовка страницы

$page\_title = "Страница товара (показ выбранного товара)";

// получаем ID товара

$id = isset($\_GET["id"]) ? $\_GET["id"] : die("ERROR: отсутствует ID.");

// подключаем файлы для работы с базой данных и файлы с объектами

include\_once "config/database.php";

include\_once "objects/product.php";

include\_once "objects/category.php";

// получаем соединение с базой данных

$database = new Database();

$db = $database->getConnection();

// подготавливаем объекты

$product = new Product($db);

$category = new Category($db);

// устанавливаем свойство ID товара для чтения

$product->id = $id;

// получаем информацию о товаре

$product->readOne();

require\_once "layout\_header.php";

?>

<!-- ссылка на все товары -->

<div class="right-button-margin">

<a href="index.php" class="btn btn-primary pull-right">

<span class="glyphicon glyphicon-list"></span> Просмотр всех товаров

</a>

</div>

<!-- HTML-таблица для отображения информации о товаре -->

<table class="table table-hover table-responsive table-bordered">

<tr>

<td>Название</td>

<td><?= $product->name; ?></td>

</tr>

<tr>

<td>Цена</td>

<td><?= $product->price; ?></td>

</tr>

<tr>

<td>Количество</td>

<td><?= $product->description; ?></td>

</tr>

<tr>

<td>Категория</td>

<td>

<?php // выводим название категории

$category->id = $product->category\_id;

$category->readName();

echo $category->name;

?>

</td>

</tr>

</table>

<?php // подвал

require\_once "layout\_footer.php";

read\_template

<?php

// форма поиска

echo "<form role='search' action='search.php'>";

echo "<div class='input-group col-md-3 pull-left margin-right-1em'>";

$search\_value = isset($search\_term) ? "value='{$search\_term}'" : "";

echo "<input type='text' class='form-control' placeholder='Поиск' name='s' required {$search\_value} />";

echo "<div class='input-group-btn'>";

echo "<button class='btn btn-primary' type='submit'><i class='glyphicon glyphicon-search'></i></button>";

echo "</div>";

echo "</div>";

echo "</form>";

// кнопка создания товара

echo "<div class='right-button-margin'>";

echo "<a href='create\_product.php' class='btn btn-primary pull-right'>";

echo "<span class='glyphicon glyphicon-plus'></span> Создать товар";

echo "</a>";

echo "</div>";

// показать товары, если они есть

if ($total\_rows > 0) {

echo "<table class='table table-hover table-responsive table-bordered'>";

echo "<tr>";

echo "<th>Товар</th>";

echo "<th>Цена</th>";

echo "<th>Количество</th>";

echo "<th>Категория</th>";

echo "<th>Действия</th>";

echo "</tr>";

while ($row = $stmt->fetch(PDO::FETCH\_ASSOC)) {

extract($row);

echo "<tr>";

echo "<td>{$name}</td>";

echo "<td>{$price}</td>";

echo "<td>{$description}</td>";

echo "<td>";

$category->id = $category\_id;

$category->readName();

echo $category->name;

echo "</td>";

echo "<td>";

// кнопка просмотра товара

echo "<a href='read\_product.php?id={$id}' class='btn btn-primary left-margin' style='margin-left: 35px'>";

echo "<span class='glyphicon glyphicon-list'></span> Просмотр";

echo "</a>";

// кнопка редактирования товара

echo "<a href='update\_product.php?id={$id}' class='btn btn-info left-margin' style='margin-left: 35px'>";

echo "<span class='glyphicon glyphicon-edit'></span> Редактировать";

echo "</a>";

// кнопка удаления товара

echo "<a delete-id='{$id}' class='btn btn-danger delete-object' style='margin-left: 35px'>";

echo "<span class='glyphicon glyphicon-remove'></span> Удалить";

echo "</a>";

echo "</td>";

echo "</tr>";

}

echo "</table>";

// пагинация

include\_once "paging.php";

}

// сообщить пользователю, что товаров нет

else {

echo "<div class='alert alert-danger' style='margin-top:70px;'>Товаров не найдено.</div>";

}

Search

<?php

// содержит переменные пагинации

include\_once "config/core.php";

// для подключения к БД и файлы с объектами

include\_once "config/database.php";

include\_once "objects/product.php";

include\_once "objects/category.php";

// создание экземпляра класса базы данных и товара

$database = new Database();

$db = $database->getConnection();

$product = new Product($db);

$category = new Category($db);

// получение поискового запроса

$search\_term = isset($\_GET["s"]) ? $\_GET["s"] : "";

$page\_title = "Вы искали \"{$search\_term}\"";

require\_once "layout\_header.php";

// запрос товаров

$stmt = $product->search($search\_term, $from\_record\_num, $records\_per\_page);

// указываем страницу, на которой используется пагинация

$page\_url = "search.php?s={$search\_term}&";

// подсчитываем общее количество строк - используется для разбивки на страницы

$total\_rows = $product->countAll\_BySearch($search\_term);

// шаблон для отображения списка товаров

include\_once "read\_template.php";

// содержит наш JavaScript и закрывающие теги html

require\_once "layout\_footer.php";

update\_product

<?php

// получаем ID редактируемого товара

$id = isset($\_GET["id"]) ? $\_GET["id"] : die("ERROR: отсутствует ID.");

// подключаем файлы для работы с базой данных и файлы с объектами

include\_once "config/database.php";

include\_once "objects/product.php";

include\_once "objects/category.php";

// получаем соединение с базой данных

$database = new Database();

$db = $database->getConnection();

// подготавливаем объекты

$product = new Product($db);

$category = new Category($db);

// устанавливаем свойство ID товара для редактирования

$product->id = $id;

// получаем информацию о редактируемом товаре

$product->readOne();

// установка заголовка страницы

$page\_title = "Редактирование товара";

include\_once "layout\_header.php";

?>

<div class="right-button-margin">

<a href="index.php" class="btn btn-default pull-right">Просмотр всех товаров</a>

</div>

<?php

// если форма была отправлена (submit)

if ($\_POST) {

// устанавливаем значения свойствам товара

$product->name = $\_POST["name"];

$product->price = $\_POST["price"];

$product->description = $\_POST["description"];

$product->category\_id = $\_POST["category\_id"];

// обновление товара

if ($product->update()) {

echo "<div class='alert alert-success alert-dismissable' style='margin-top: 70px'>";

echo "Товар был обновлён";

echo "</div>";

}

// если не удается обновить товар, сообщим об этом пользователю

else {

echo "<div class='alert alert-danger alert-dismissable' style='margin-top: 70px'>";

echo "Невозможно обновить товар.";

echo "</div>";

}

}

?>

<form action="<?= htmlspecialchars($\_SERVER["PHP\_SELF"] . "?id={$id}"); ?>" method="post">

<table class="table table-hover table-responsive table-bordered">

<tr>

<td>Название</td>

<td><input type="text" name="name" value="<?= $product->name; ?>" class="form-control" /></td>

</tr>

<tr>

<td>Цена</td>

<td><input type="text" name="price" value="<?= $product->price; ?>" class="form-control" /></td>

</tr>

<tr>

<td>Количество</td>

<td><input type="text" name="description" value="<?= $product->description; ?>" class="form-control" /></td>

</tr>

<tr>

<td>Категория</td>

<td>

<?php

$stmt = $category->read();

// помещаем категории в выпадающий список

echo "<select class='form-control' name='category\_id'>";

echo "<option selected disabled value=''>Выберите категорию</option>";

while ($row\_category = $stmt->fetch(PDO::FETCH\_ASSOC)) {

$category\_id = $row\_category["id"];

$category\_name = $row\_category["name"];

// необходимо выбрать текущую категорию товара

if ($product->category\_id == $category\_id) {

echo "<option value='$category\_id' selected>";

} else {

echo "<option value='$category\_id'>";

}

echo "$category\_name</option>";

}

echo "</select>";

?>

</td>

</tr>

<tr>

<td></td>

<td>

<button type="submit" class="btn btn-primary">Обновить</button>

</td>

</tr>

</table>

</form>

<?php // подвал

require\_once "layout\_footer.php";