Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационных компьютерных систем

Дисциплина "Объектно-ориентированное программирование"

К защите допустить:
Руководитель курсовой работы
старший преподаватель
кафедры
_____ А.В.Михалькевич
07.07.2025

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Интернет-магазин телефонов

БГУИР КР 1-40 05 01-10 № 140 ПЗ

Студент (подпись студента)

Курсовая работа
представлена на проверку
07.07.2025
_____
(подпись студента)

2025

Реферат

БГУИР КР 1-40 05 01-10 № 140 ПЗ, гр. 814302

, Интернет-магазин телефонов, Минск: БГУИР - 2025.

Пояснительная записка 59181 с., 16 рис., 0 табл.

Ключевые слова: сайт

Предмет Объектно-ориентированное программирование, А.В.Михалькевич

В нашем мире Интернет является главной платформой для связи с потребителями, он облегчает жизнь многим. Но так же немаловажно, что Интернет все более и более упрощен и доступен для поиска важной информации. Сейчас много внимания уделяется разработке различных сайтов, программ, которые в дальнейшем будут использованы людьми для просмотра новостей или получения нужной информации. Следует учитывать тот факт, что в наше время делать покупки через Интернет уже становится обычным делом для каждого из нас. Множество людей по всему миру ежедневно оформляют покупки через интернет-магазины. Создано множество сайтов для покупки товаров на ваш вкус и взгляд. И вопрос состоит в том, каким сервисом лучше всего воспользоваться? Конечно, каждый производитель пытается как-нибудь заманить клиента к себе. Но если над вашим запросом работает целая команда профессионалов, результат не заставит себя долго ждать. Клиент сам захочет получить самый качественный товар и возможность индивидуального подхода. В курсовой работе поставлена цель: проектирование сайта интернет-магазина телефонов. Для достижения поставленной цели необходимо решить следующие задачи: 1 Изучение современных инструментов разработки сайтов. 2 Разработка макета интернет-магазина. 3 Создание баз данных для хранения информации. 4 Интеграция сервиса для работы с картами. 5 Организация автоматической загрузки информации, запрашиваемой пользователем. Данный сайт предназначен для людей, которые желают приобрести себе самый качественный товар, при этом получить консультацию у профессионалов.

-

Содержание

Введение
1 Описание проекта
2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ
3 ИНСТРУМЕНТАРИЙ
4 АРХИТЕКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ MVC
5 РАЗРАБОТКА БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ
6 ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА
Заключение
Список использованных источников
Приложения

Введение

В нашем мире Интернет является главной платформой для связи с потребителями, он облегчает жизнь многим. Но так же немаловажно, что Интернет все более и более упрощен и доступен для поиска важной информации. Сейчас много внимания уделяется разработке различных сайтов, программ, которые в дальнейшем будут использованы людьми для просмотра новостей или получения нужной информации. Следует учитывать тот факт, что в наше время делать покупки через Интернет уже становится обычным делом для каждого из нас. Множество людей по всему миру ежедневно оформляют покупки через интернет-магазины. Создано множество сайтов для покупки товаров на ваш вкус и взгляд. И вопрос состоит в том, каким сервисом лучше всего воспользоваться? Конечно, каждый производитель пытается как-нибудь заманить клиента к себе. Но если над вашим запросом работает целая команда профессионалов, результат не заставит себя долго ждать. Клиент сам захочет получить самый качественный товар и возможность индивидуального подхода. В курсовой работе поставлена цель: проектирование сайта интернет-магазина телефонов. Для достижения поставленной цели необходимо решить следующие задачи: 1 Изучение современных инструментов разработки сайтов. 2 Разработка макета интернет-магазина. 3 Создание баз данных для хранения информации. 4 Интеграция сервиса для работы с картами. 5 Организация автоматической загрузки информации, запрашиваемой пользователем. Данный сайт предназначен для людей, которые желают приобрести себе самый качественный товар, при этом получить консультацию у профессионалов.

1 Описание проекта

Курсовой проект представляет собой сайт интернет-магазина телефонов с возможностью просмотра и выбора товаров, просмотра последних новостей.

Создана возможность навигации по сайту. Для статей и фото отображается ссылка для перехода на источник.

Сайт обеспечивает пользователем уникальную возможность связи с нашей командой при выборе себе покупки.

На основной странице сайта представлена возможность просмотра товара, связи с командой, описание и возможности компании, новости, возможность просмотра местонахождения на карте, просмотр отзывов, просмотр новых проектов.

В каждом разделе имеется подробная и последняя информация о товаре.

Сайт разработан при помощи языка HTML, с использованием языка CSS, а также языка сценариев JavaScript[1], [2].

Сайт реализован на русском языке.

Навигация по сайту довольно проста и интуитивно понятна, использование меню дает представление о сайте.

2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ

Наследование — концепция объектно-ориентированного программирования, согласно которой абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения.

Наследование – это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью.

Класс, от которого производится наследование, называется базовым или родительским. Новый класс – потомком, наследником или производным классом.

Необходимо отметить, что производный класс полностью удовлетворяет спецификации родительского, однако может иметь дополнительную функциональность.

С точки зрения интерфейсов, каждый производный класс полностью реализует интерфейс родительского класса. Обратное не верно.

В объектно-ориентированном программировании абстрактные типы данных называются классами.

Суперкласс — класс, производящий наследование в подклассах, т. е. класс, от которого наследуются другие классы. Суперклассом может быть подкласс, базовый класс, абстрактный класс и интерфейс.

Подкласс — класс, наследуемый от суперкласса или интерфейса, т. е. класс определённый через наследование от другого класса или нескольких таких классов. Подклассом может быть суперкласс.

Базовый класс — это класс, находящийся на вершине иерархии наследования классов и в основании дерева подклассов, т. е. не являющийся подклассом и не имеющий наследований от других суперклассов или интерфейсов. Базовым классом может быть абстрактный класс и интерфейс. Любой не базовый класс является подклассом.

Интерфейс — это структура, определяющая чистый интерфейс класса, состоящий из абстрактных методов. Интерфейсы участвуют в иерархии наследований классов и интерфейсов.

Суперинтерфейс — это аналог суперкласса в иерархии наследований, т. е. это интерфейс производящий наследование в подклассах и подинтерфейсах.

Интерфейс-потомок — это аналог подкласса в иерархии наследований интерфейсов, т. е. это интерфейс наследуемый от одного или нескольких суперинтерфейсов.

Базовый интерфейс — это аналог базового класса в иерархии

наследований интерфейсов, т. е. это интерфейс, находящийся на вершине иерархии наследования.

Иерархия наследования — дерево, элементами которого являются классы и интерфейсы.

Наследование является механизмом повторного использования кода и способствует независимому расширению программного обеспечения через открытые классы и интерфейсы. Установка отношения наследования между классами порождает иерархию классов.

Простое наследование — «Простое» наследование, иногда называемое одиночным наследованием, описывает родство между двумя классами: один из которых наследует второму. Из одного класса могут выводится многие классы, но даже в этом случае подобный вид взаимосвязи остается «простым» наследованием.

Множественное наследование — при множественном наследовании, у класса может быть более одного предка. В этом случае класс наследует методы всех предков. Достоинства такого подхода в большей гибкости.

Множественное наследование — потенциальный источник ошибок, которые могут возникнуть из-за наличия одинаковых имён методов в предках. Практически всегда можно обойтись без использования данного механизма. Однако, если такая необходимость всё-таки возникла, то для разрешения конфликтов использования наследованных методов с одинаковыми именами возможно, например, применить операцию расширения видимости — «::» — для вызова конкретного метода конкретного родителя.

В «JavaScript» используется прототипное наследование.

3 ИНСТРУМЕНТАРИЙ

3.1 Обоснование используемых инструментов

Brackets — свободный текстовый редактор для веб-разработчиков. Brackets ориентирован на работу с HTML, CSS и JavaScript. Эти же технологии лежат в основе самого редактора, что обеспечивает его кроссплатформенность т. е. совместимость с операционными системами Mac, Windows и Linux. Brackets создан и развивается Adobe Systems под лицензией MIT License и поддерживается на GitHub.

Положительные отличия редактора Brackets:

- русскоязычный интерфейс. В Sublime всё на английском – это не критично, но всё-таки создаёт дискомфорт.
- простая установка плагинов и тем оформления. Sublime Text в этом плане немного заморочен, а в Brackets это реализовано более удобно. К тому же, перед установкой плагина можно почитать его краткое описание.
- предпросмотр результата в браузере встроен. Вы пишете код, и тут же видите изменения в браузере. Прямая связь.

- редактор Brackets создан компанией Adobe, распространяется бесплатно и поддерживается большим сообществом. То есть постоянно развивается и улучшается.
- наконец, Brackets сам написан на HTML, CSS и JavaScript, а потому прекрасно работает не только на Windows, но и на Mac, и на Linux.

Notepad++ — текстовый редактор, предназначенный для программистов и всех тех, кого не устраивает скромная функциональность входящего в состав Windows Блокнота.

Основные особенности программы:

- подсветка текста и возможность сворачивания блоков, согласно синтаксису языка программирования;
- поддержка большого количества языков (C, C++, Java, XML, HTML, PHP, *Java Script*, ASCII, VB/VBS, SQL, CSS, Pascal, Perl, Python, Lua, TCL, Assembler);
- WYSIWYG (печатаешь и получаешь то, что видишь на экране);
- настраиваемый пользователем режим подсветки синтаксиса;
- авто-завершение набираемого слова;
- одновременная работа с множеством документов;
- одновременный просмотр нескольких документов;
- поддержка регулярных выражений Поиска/Замены;
- полная поддержка перетягивания фрагментов текста;
- динамическое изменение окон просмотра;
- автоматическое определение состояния файла;
- увеличение и уменьшение;
- заметки;
- выделение скобок при редактировании текста;
- запись макроса и его выполнение.

3.2 Использование системы контроля версий GIT

Система контроля версий (СКВ) – это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов [4].

СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, испортить или потерять файлы, всё можно будет легко восстановить.

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

Чтобы решить эту проблему, программисты уже давно разработали локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (рисунок 1).

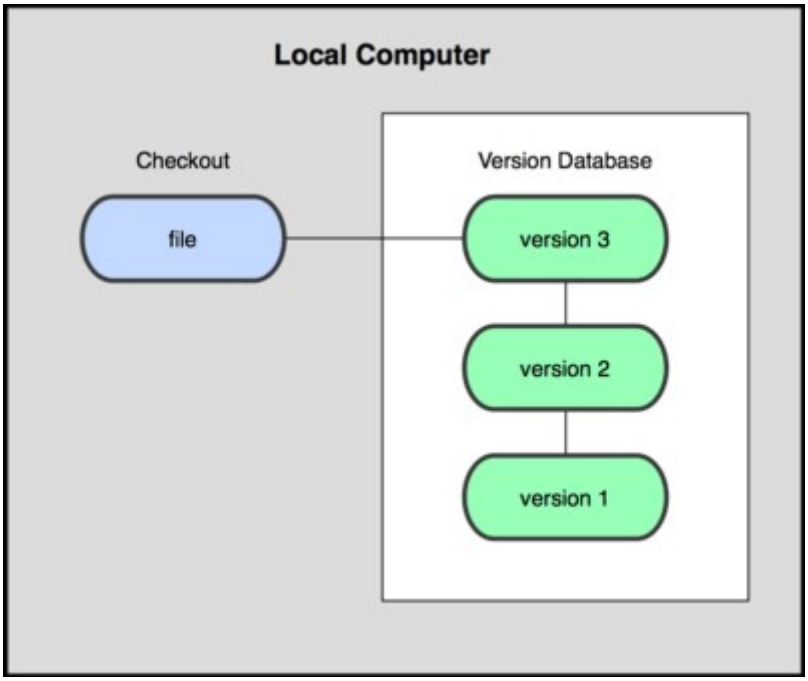

Рисунок 1- Схема локальной СКВ

Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч – файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (рисунок 2).

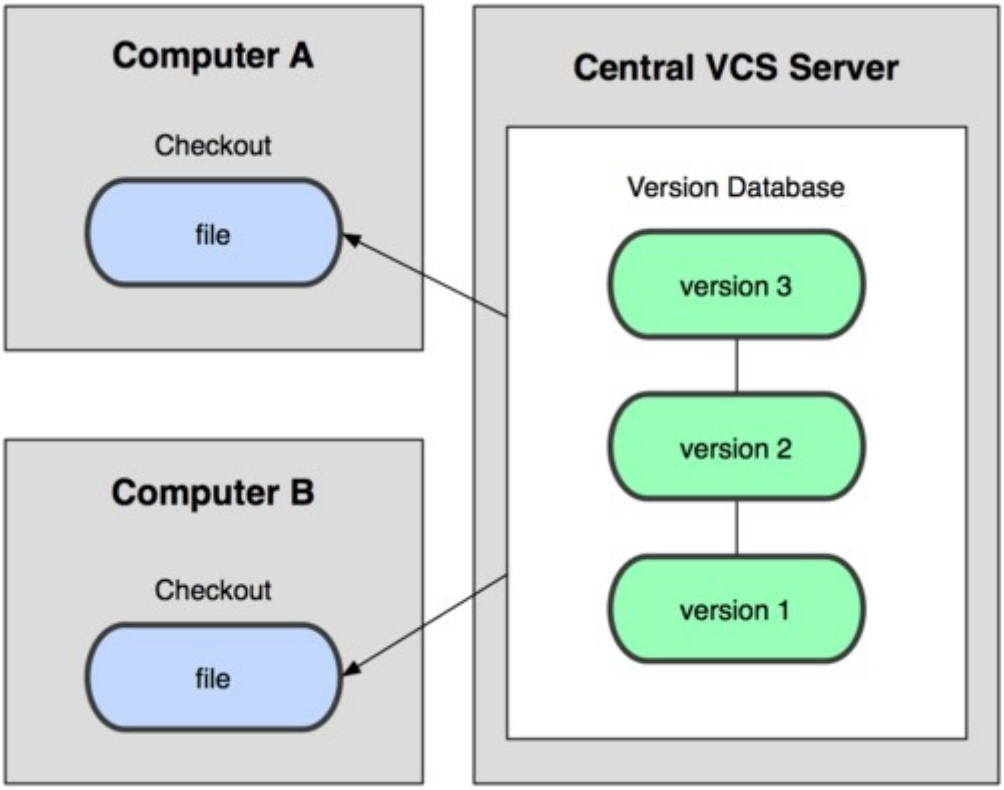

Рисунок 2 - Схема централизованного контроля версий

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный – централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё – всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

И в этой ситуации в игру вступают распределённые системы контроля версий (РСКВ). В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (рисунок 3).

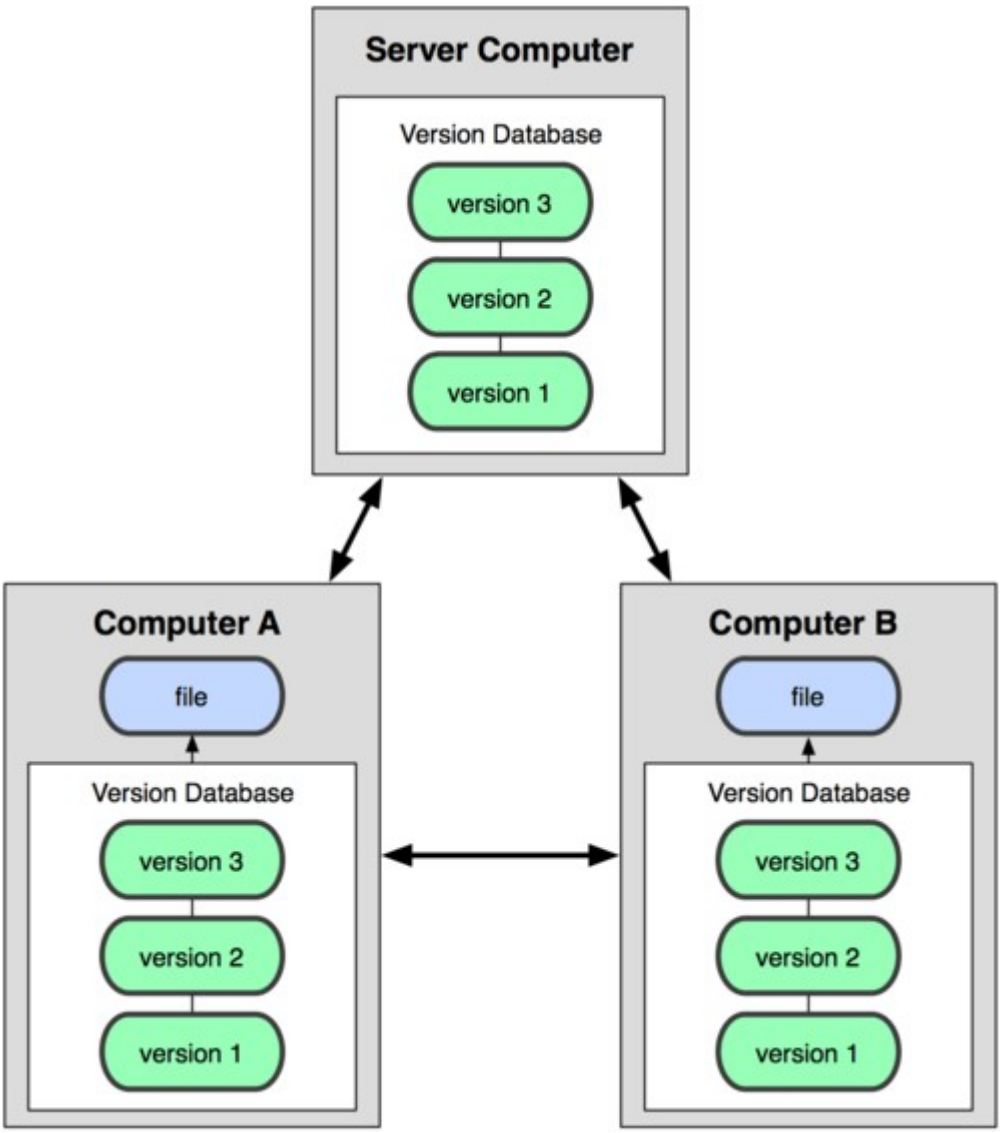

Рисунок 3 – Схема распределённой системы контроля версий

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

В нашем курсовом проекте использовался онлайн-репозиторий GitHub. Ссылка на него: https://github.com/VladNesterov753/Kursach.git

4 АРХИТЕКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ MVC

Шаблон проектирования Модель – Представление – Контроллер (MVC) – это шаблон программной архитектуры, построенный на основе сохранения представления данных отдельно от методов, которые взаимодействуют с данными [5].

Вид — эта часть отвечает за вывод информации на экран приложения с минимальным количеством логики. То есть попросту говоря, этот блок отвечает за внешний вид нашего

приложения. Задача представления хранить дизайн скрипта.

Контроллер — блок, который получает данные от пользователя, обрабатывает, нормализует их, также выполняет проверку правильности ввода и передает эти обработанные данные в нужную модель. Также он принимает данные от модели, затем выбирает нужное представление, наполняет его данными и отображает на экране браузера. Но при этом, еще раз уточню, контроллер не должен содержать в себе никакой информации о внешнем виде веб-приложения. Контроллер можно рассмотреть как связующее звено между представлением и моделью.

Модель — это основа логики, которая она отвечает за расчеты, выборку информации из базы данных, изменение информации в БД и т.д. Модель можно представить как библиотеку различных функций, позволяющих реализовывать функционал нашего приложения. Это блок, который получает данные от контроллера, далее на основе этих данных производит необходимые преобразования, либо опять же выбирает данные из БД или изменяет их, а затем передает результат своей работы назад контроллеру.

Не смотря на то, что схема MVC была первоначально разработана для персональных компьютеров, она была адаптирована и широко используется веб-разработчиками из-за точного разграничения задач и возможности повторного использования кода. Схема стимулирует развитие модульных систем, что позволяет разработчикам быстро обновлять, добавлять или удалять функционал.

Название шаблона проектирования определяется тремя его основными составляющими частями: Модель, Представление и Контроллер. Визуальное представление шаблона MVC выглядит, как показано на приведенной ниже диаграмме (рисунок 4):

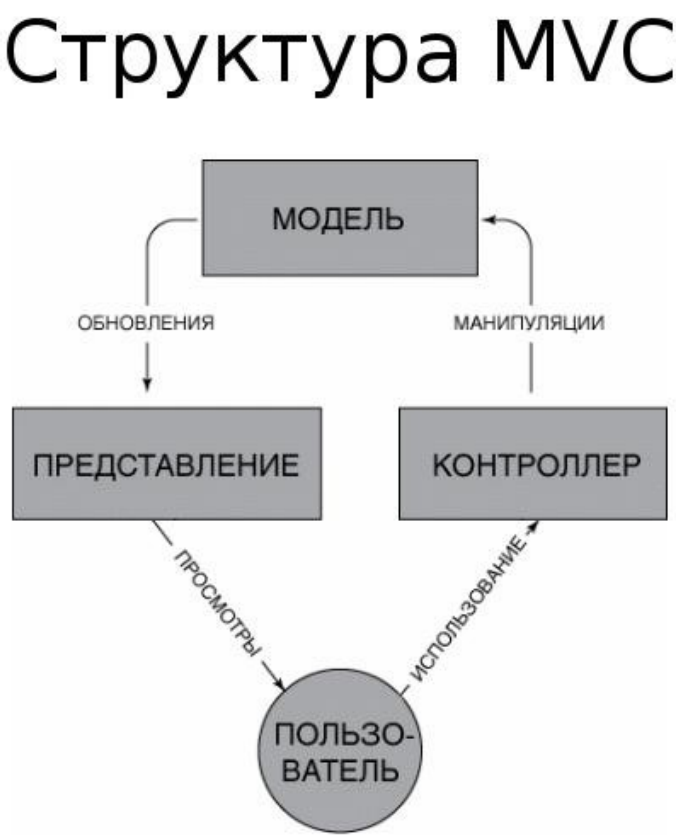

Рисунок 4 - Схема шаблона MVC

На рисунке показана структура одностороннего потока данных и пути его следования между различными компонентами, а также их взаимодействие.

Моделью называют постоянное хранилище данных, используемых во всей структуре. Она должна обеспечивать доступ к данным для их просмотра, отбора или записи. В общей структуре Модель является мостом между компонентами Представление и Контроллер.

При этом Модель не имеет никакой связи или информации о том, что происходит с данными, когда они передаются компонентам Представление или Контроллер. Единственная задача Модели – обработка данных в постоянном хранилище, поиск и подготовка данных, передаваемых другим составляющим MVC.

Модель должна выступать в качестве «привратника», стоящего возле хранилища данных и не задающего вопросов, но принимающего все поступающие запросы. Зачастую это наиболее сложная часть системы MVC. Компонент Модель – это вершина всей структуры, так как без нее невозможна связь между Контроллером и Представлением.

Представление – это часть системы, в которой данным, запрашиваемым у Модели, задается окончательный вид их вывода. В веб-приложениях, созданных на основе MVC, Представление – это компонент, в котором генерируется и отображается HTML-код.

Представление также перехватывает действие пользователя, которое затем передается

Контроллеру. Характерным примером этого является кнопка, генерируемая Представлением. Когда пользователь нажимает ее, запускается действие в Контроллере.

Существует несколько распространенных заблуждений относительно компонента Представление. Например, многие ошибочно полагают, что Представление не имеет никакой связи с Моделью, а все отображаемые данные передаются от Контроллера. В действительности такая схема потока данных не учитывает теорию, лежащую в основе MVC архитектуры.

Кроме этого определение Представления как файла шаблона также является неточным. Но это не вина одного человека, а результат множества ошибок различных разработчиков, которые приводят общему заблуждению. После чего они неправильно объясняют это другим. На самом деле Представление это намного больше, чем просто шаблон. Но современные MVC-ориентированные фреймворки до такой степени впитали этот подход, что никто уже не заботится о том, поддерживается ли верная структура MVC или нет.

Компоненту Представление никогда не передаются данные непосредственно Контроллером. Между Представлением и Контроллером нет прямой связи – они соединяются с помощью Модели.

Его задача заключается в обработке данных, которые пользователь вводит и обновлении Модели. Это единственная часть схемы, для которой необходимо взаимодействие пользователя.

Контроллер можно определить, как сборщик информации, которая затем передается в Модель с последующей организацией для хранения. Он не содержит никакой другой логики, кроме необходимости собрать входящие данные. Контроллер также подключается только к одному Представлению и одной Модели. Это создает систему с односторонним потоком данных с одним входом и одним выходом в точках обмена данными.

Контроллер получает задачи на выполнение только когда пользователь взаимодействует с Представлением, и каждая функция зависит от взаимодействия пользователя с Представлением. Наиболее распространенная ошибка разработчиков заключается в том, что они путают Контроллер со шлюзом, поэтому присваивают ему функции и задачи, которые относятся к Представлению.

Также распространенной ошибкой является наделение Контроллера функциями, которые отвечают только за обработку и передачу данных из Модели в Представление. Но согласно структуре MVC паттерна это взаимодействие должно осуществляться между Моделью и Представлением.

5 РАЗРАБОТКА БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ

Каждый владелец сайта знает, что для правильного функционирования сайта нужны не только файлы с кодом страниц, но и базы данных. Для взаимодействия с базами данных используются системы управления базами данных (СУБД).

База данных представляет собой определенный набор данных, которые, как правило, связаны объединяющим признаком либо свойством (или несколькими). Эти данные

упорядочены, например, по алфавиту. Обилие различных данных, которые могут быть помещены в единую базу, ведет к множеству вариаций того, что может быть записано: личные данные пользователей, записи, даты, заказы и так далее.

В первую очередь это удобно тем, что информацию можно быстро заносить в базу данных и так же быстро ее извлекать при необходимости. Если на заре развития web-разработки все необходимые данные нужно было прописывать в коде страницы, то теперь такая необходимость отсутствует – нужная информация может быть запрошена из базы данных при помощи скриптов. Специальные алгоритмы хранения и поиска информации, которые используются в базах данных, позволяют находить нужные сведения буквально за доли секунд – а при работе в виртуальном пространстве скорость работы ресурса важна как ничто другое.

Немаловажной является и взаимосвязь информации в базе данных: изменение одной строчки может привести к значительным изменениям других строк. Работать с данными таким образом гораздо проще и быстрее, чем, если бы изменения касались только одного места в базе данных.

Как можно догадаться уже из названия, система управления базами данных (или сокращенно СУБД) представляет собой программное обеспечение, которое используется для создания и работы с базами данных. Главная функция СУБД – это управление данными (которые могут быть как во внешней, так и в оперативной памяти). СУБД обязательно поддерживает языки баз данных, а также отвечает за копирование и восстановление данных после каких-либо сбоев.

Что касается классификации баз данных, то тут возможны различные варианты. К примеру, можно разделить базы по модели данных: иерархические (имеют древовидную структуру), сетевые (по своей структуре похожи на иерархические), реляционные (используются для управления реляционными базами данных), объектно-ориентированные (используются для объектной модели данных) и объектно-реляционные (некое слияние реляционного и объектно-ориентированного вида баз данных).

Либо, если деление идет по тому, где размещается СУБД, их можно разделить на локальные – вся СУБД размещается на одном компьютере, и распределенные – части системы управления базами данных находятся на нескольких компьютерах.

Файл-серверные, клиент-серверные и встраиваемые – такие названия носят СУБД, если разделить их по способу доступа к базам данных. Файл-серверные СУБД на данный момент уже считаются устаревшими; в основном идет использование клиент-серверных (СУБД, которые располагаются на сервере вместе с самой базой данных) и встраиваемых (не требующих отдельной установки) систем.

Информация, которая хранится в базах данных, не ограничивается только текстовыми или графическими файлами – современные версии СУБД поддерживают также форматы аудио и видеофайлов.

Зачем же нужны эти СУБД? Помимо основной своей функции – хранения и систематизации огромного количества информации – они позволяют быстро обрабатывать клиентские запросы и выдавать свежую и актуальную информацию.

Реляционные и объектно-реляционные СУБД являются одними из самых распространенных

систем. Они представляют собой таблицы, у которых каждый столбец (который называется “field” или «поле») упорядочен и имеет определенное уникальное название. Последовательность строк (их называют “records” или «записи») определяется последовательностью ввода информации в таблицу. При этом обрабатывание столбцов и строк может происходить в любом порядке. Таблицы с данными связаны между собой специальными отношениями, благодаря чему с данными из разных таблиц можно работать – к примеру, объединять их – при помощи одного запроса.

Для управления реляционными базами данных применяется особый язык программирования – SQL. Сокращение расшифровывается как “Structured query language”, в переводе на русский «язык структурированных запросов».

Команды, которые используются в SQL, делятся на те, которые манипулируют данными, те, которые определяют данные, и те, которые управляют данными.

Схема работы с базой данных представлена на рисунке 5:

Рисунок 5 - Схема работы с базой данных

6 ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

При загрузке сайта, пользователь попадает на главную страницу, где отображено меню.

Сайт реализован с интуитивно понятной навигацией, которая находится в правом верхнем углу (рисунок 6).

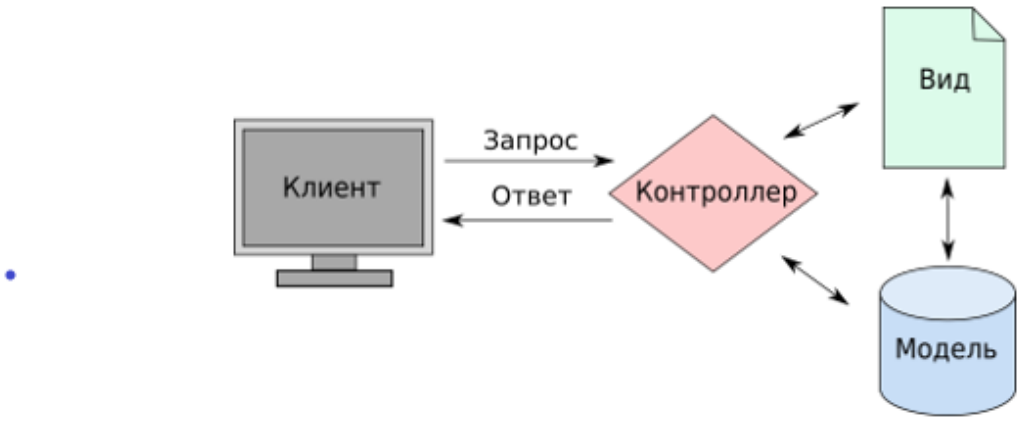

Рисунок 6 – Навигация

Чтобы выбрать пункт из меню, достаточно навести курсор. Выбранный пункт выделится желтым цветом с нижним подчеркиванием. Пример показан на рисунке 7:

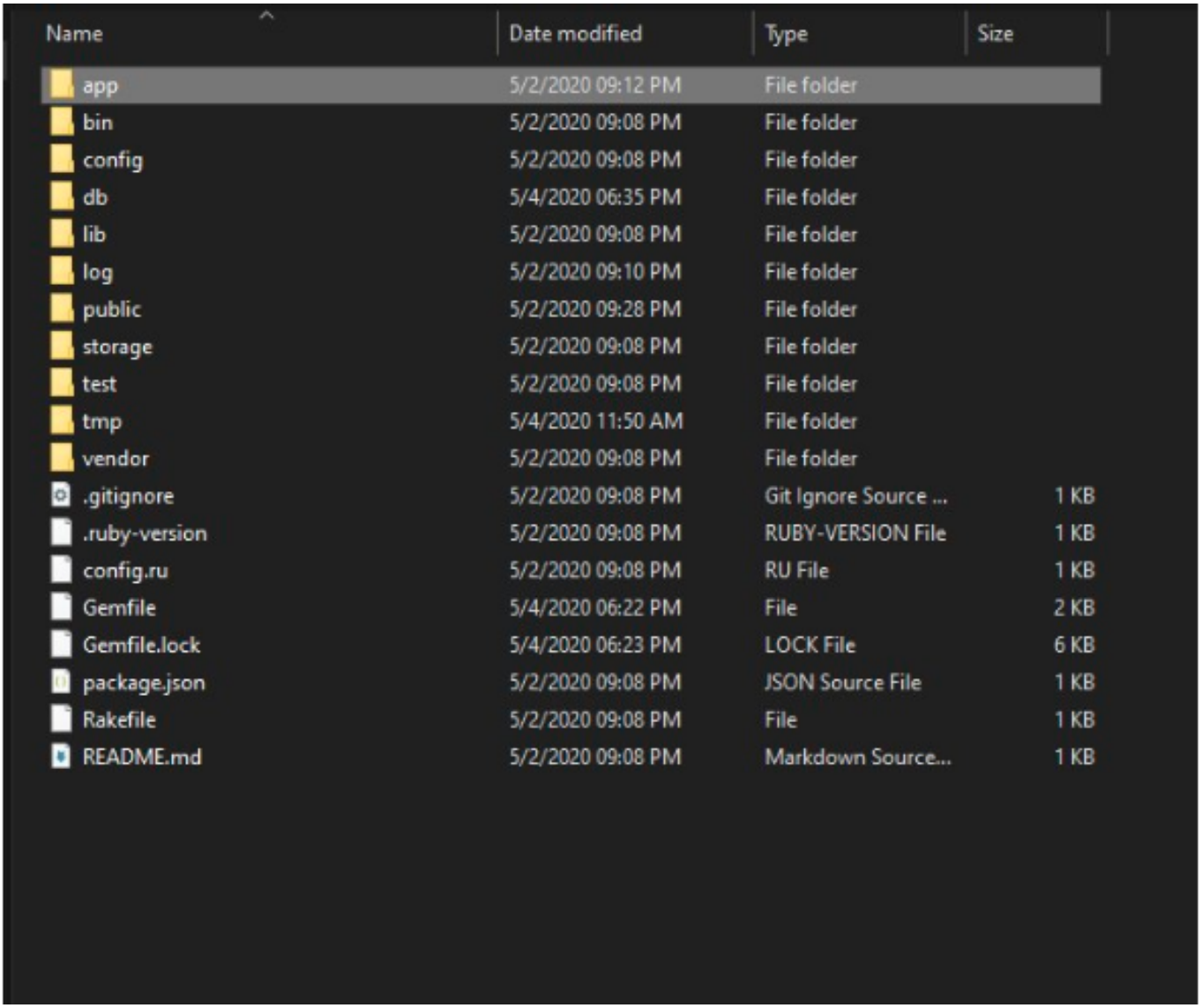

Рисунок 7 – Выбор пункта из меню

Под главным заголовком находится одна из кнопок главного меню. При наведении она подсвечивается белым цветом (рисунок 8).

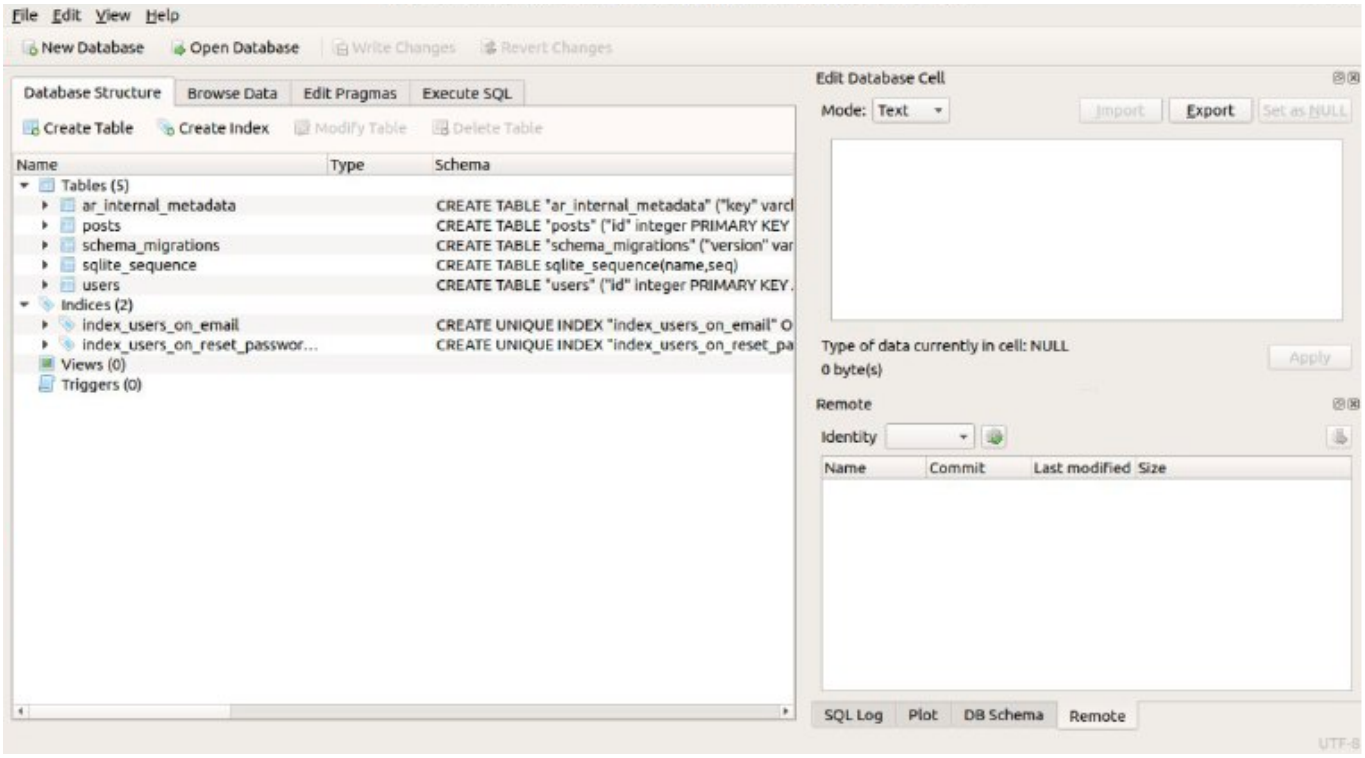

Рисунок 8 – Выбор меню

При выборе пунктов из меню нас переносит вниз к подробной информации. На рисунке изображен пункт “О нас” (рисунок 9).

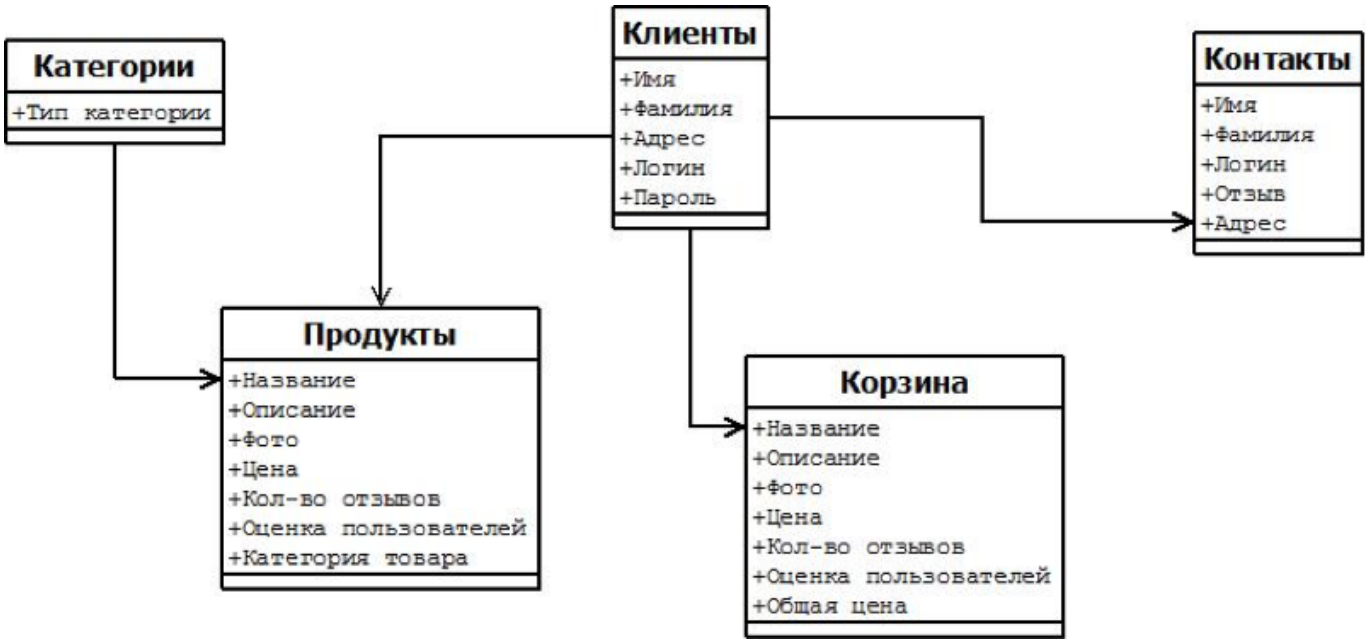

Рисунок 9 – Пункт “О нас”

Как видно из рисунка, на всем сайте нас будет сопровождать шапка с пунктами меню, при нажатии на шапке логотипа “MoGo”, вас перенесет обратно на главную страницу. Так же из рисунка видно, что на картинки добавлен эффект градиента с выплывающим текстом.

При нажатии в меню пункта “Каталог”, вас перенесет так как показано на картинке(рисунок 10). В правой части реализован аккордеон, при нажатии на стрелочки, текст выплывает и исчезает.

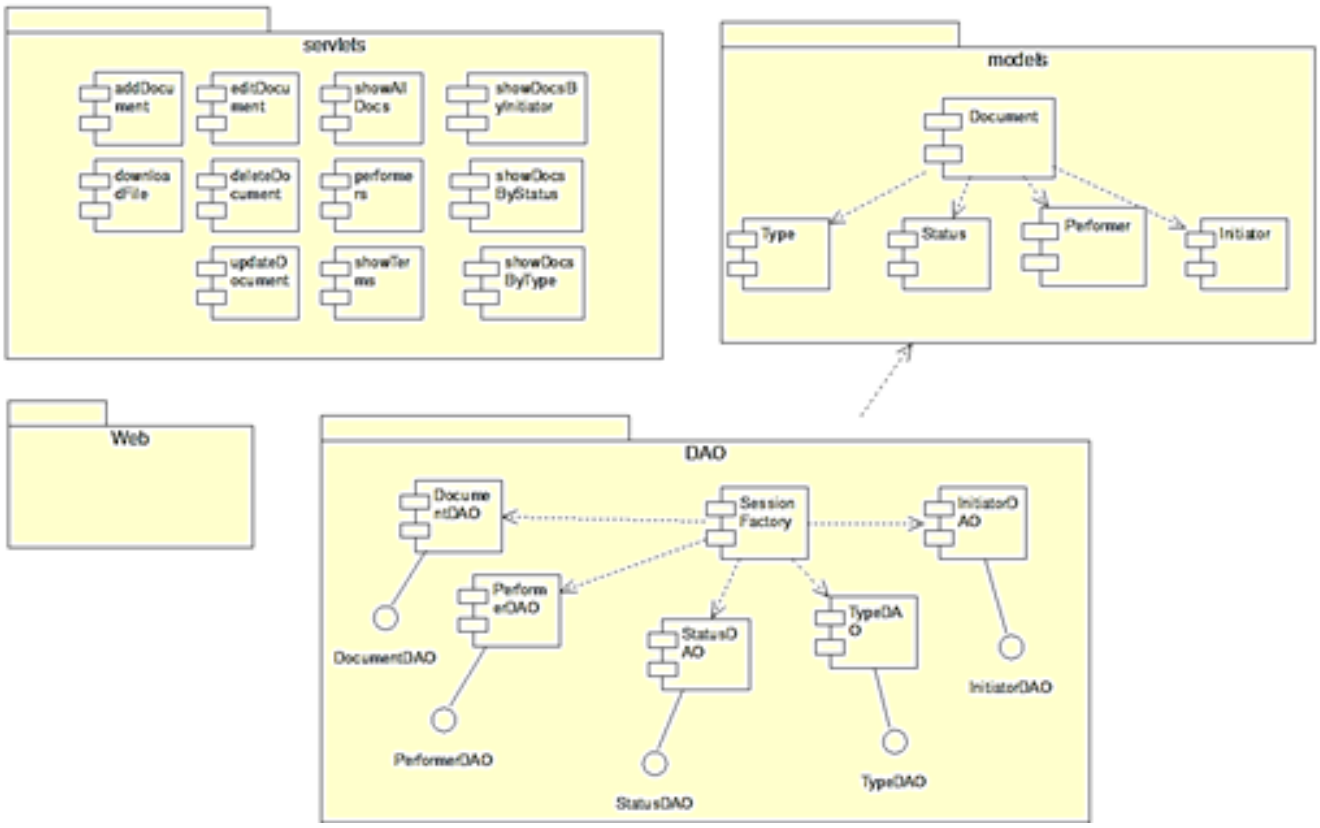

Рисунок 10 – Вкладка “Каталог”

При выборе пункта “Наш блог” (рисунок 11), вас перенесет к новостям. При нажатии на текст или картинку, вас перенесет на страницу в Instagram.

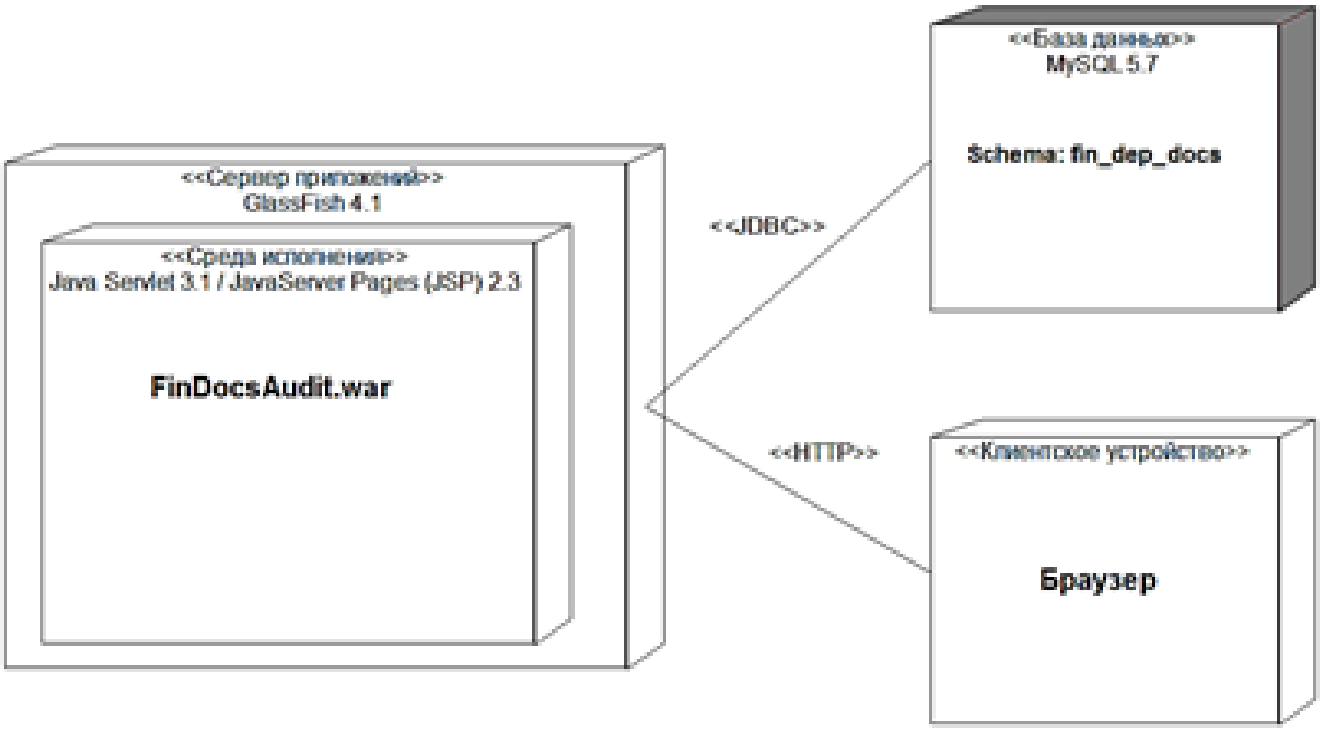

Рисунок 11 – Вкладка “Наш блог”

Так же вы можете открыть карту, как на рисунке 12.

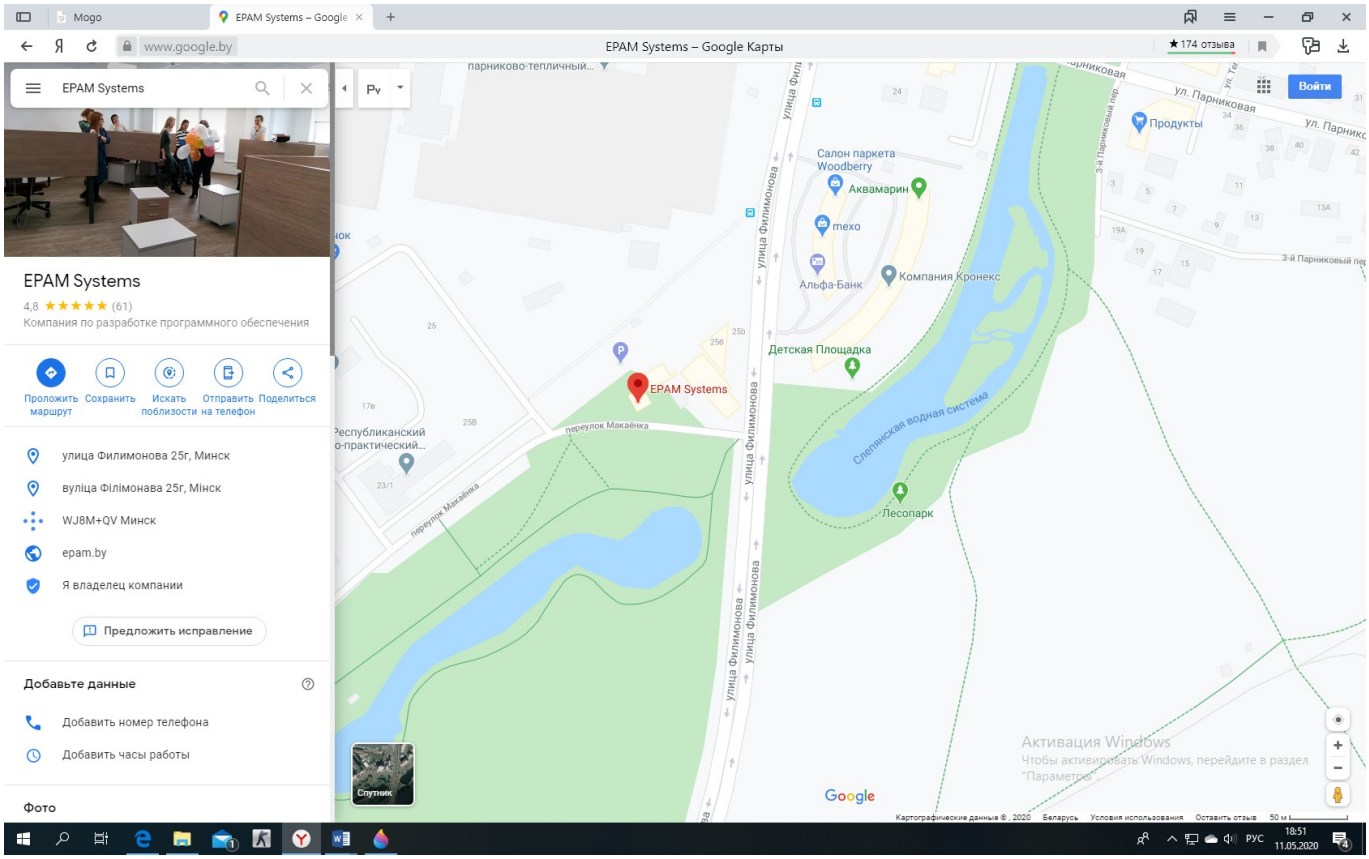

Рисунок 12 – Карта

При выборе пункта “Наши проекты” (рисунок 13), вас перенесет к списку товаров. Как видно из рисунка, при наведении на картинку она подсвечивается градиентом и выплывает текст.

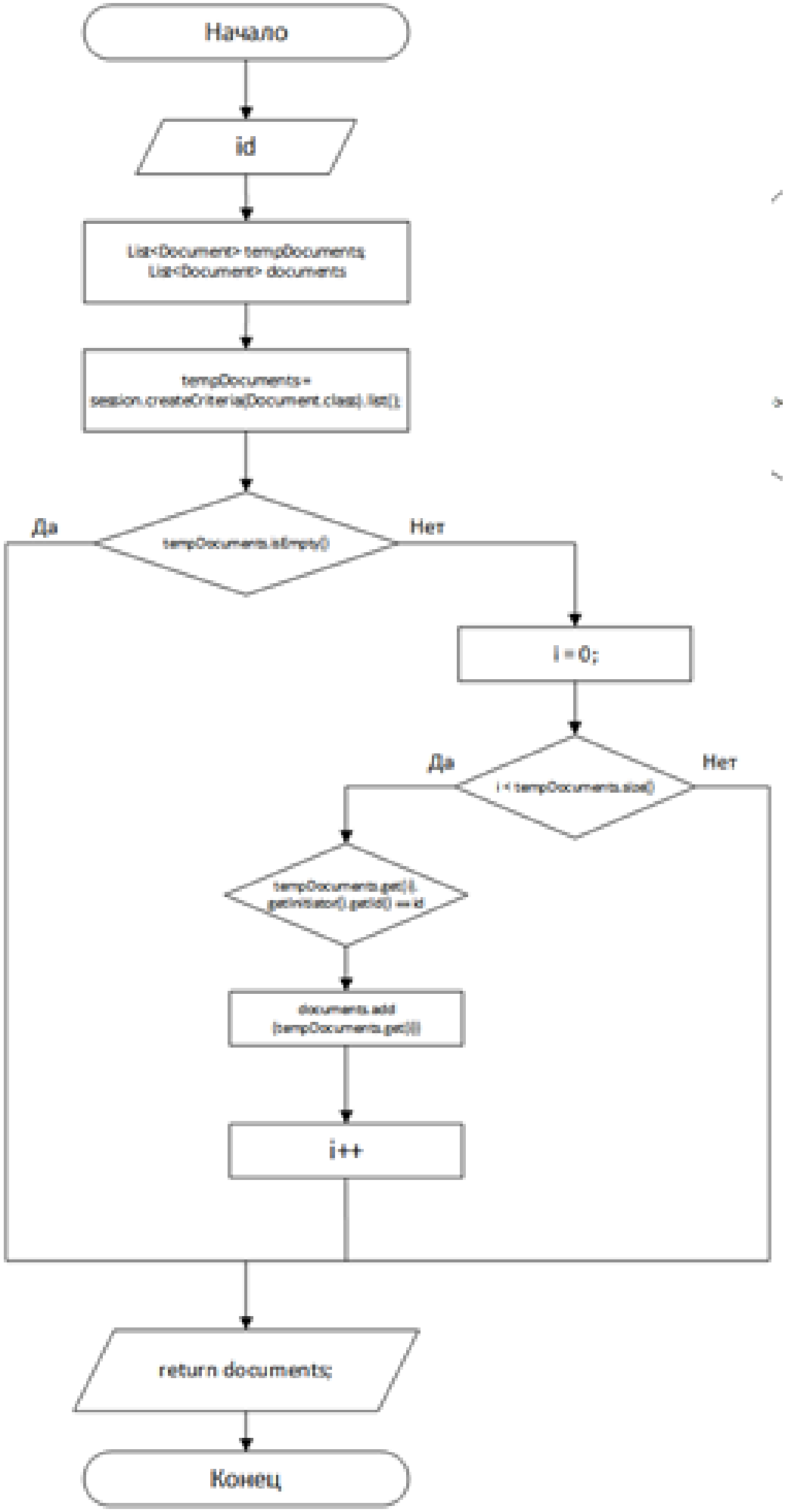
Начало
id
List<Document> tempDocuments;
List<Document> documents
tempDocuments =
session.createCriteria(Document.class).list();
Да
tempDocuments.isEmpty()
Нет
i = 0;
Да
i < tempDocuments.size()
Нет
tempDocuments.get(i).
getInstance().getId() == id
documents.add
(tempDocuments.get(i))
i++
return documents;
Конец

Рисунок 13 – Вкладка “Наши проекты”

Выбор пункта “Контакты” (рисунок 14). Здесь вы можете видеть наши контакты в Instagram, Twitter, Facebook.

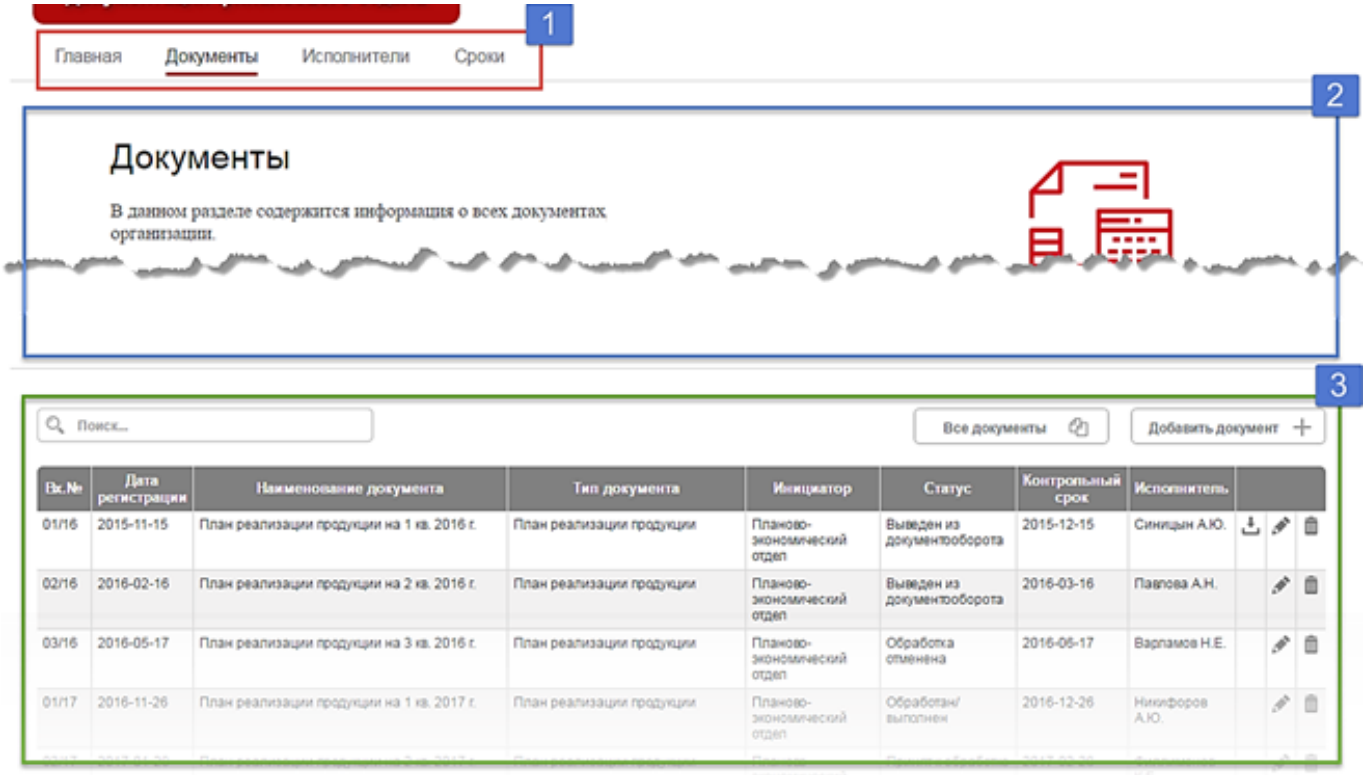

Рисунок 14 – Вкладка “Контакты”

При выборе пункта корзины, вас перенесет к нашей команде, которая проведет консультацию и оформит заказ (рисунок 15). Фотографии, при наведении, подсвечиваются градиентом и всплывают ссылки на социальные сети наших работников.

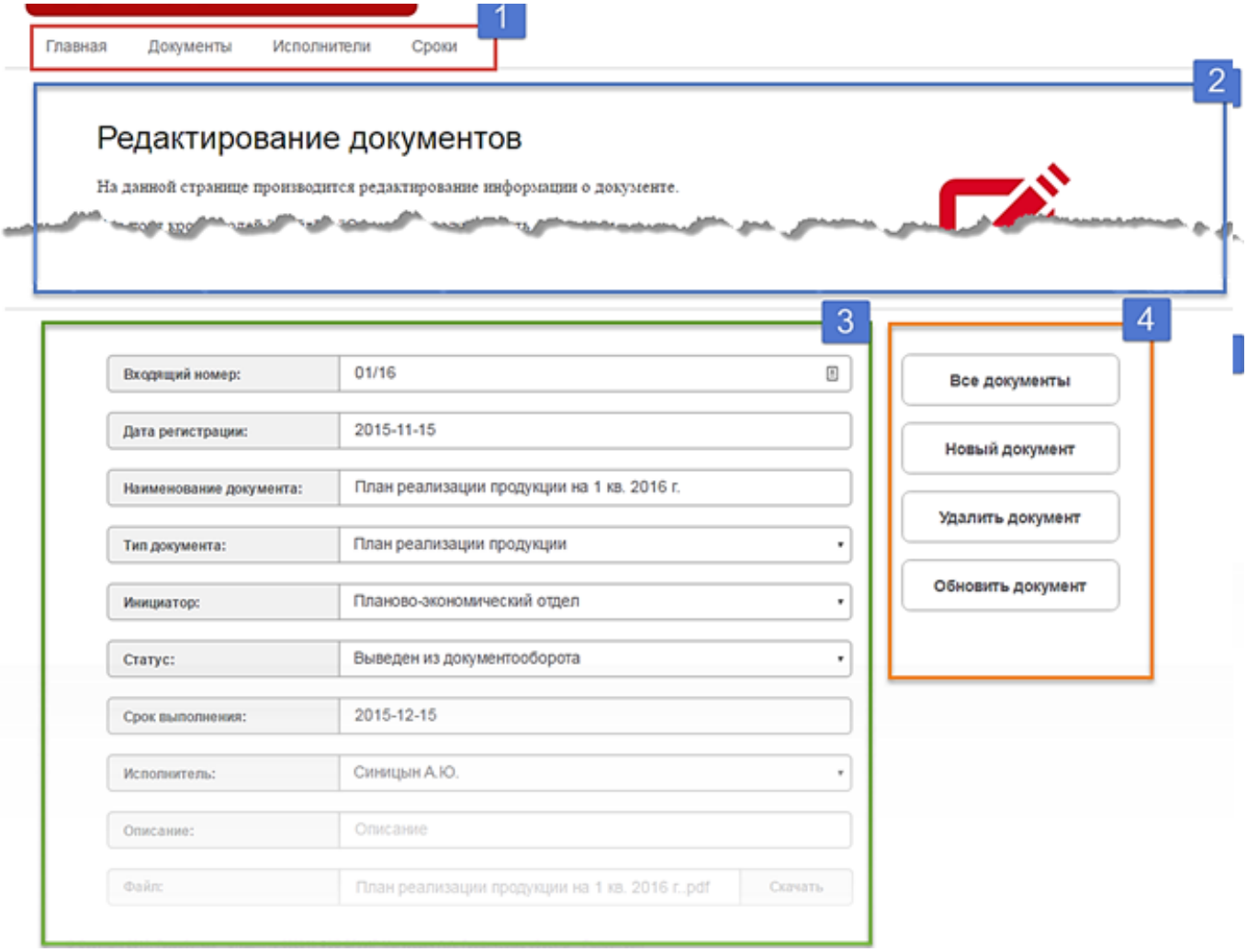

Рисунок 15 – Корзина

При выборе пункта поиска, вас перенесет к основным параметрам поиска с подходящими моделями. Здесь реализован горизонтальный скролл (рисунок 16).

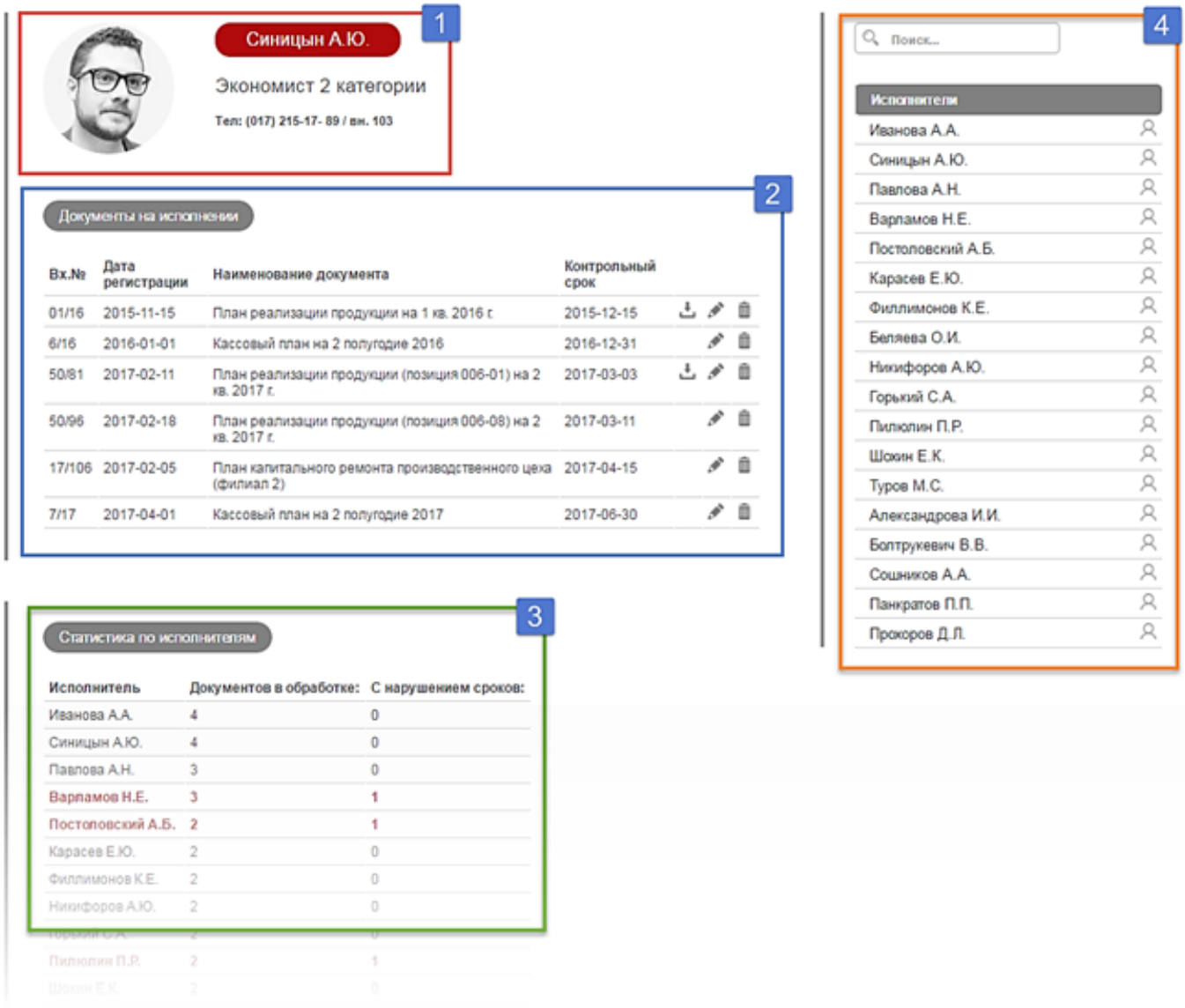

Рисунок 16 – Вкладка поиска

Заключение

В современном мире создание сайтов интернет-магазинов играет огромную роль на IT-пространстве. Эта необходимость связана с развитием информационных технологий, которые обусловлены ростом объема информации, а также значением Интернета в жизни человека. В настоящее время, использование различных интернет-магазинов вошло в привычную жизнь каждого из нас. Интернет-магазин — сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или через мобильное приложение, сформировать заказ на покупку, выбрать способ оплаты и доставки заказа, оплатить заказ. При этом продажа товаров осуществляется дистанционным способом и она накладывает ограничения на продаваемые товары. Типичный интернет-магазин позволяет клиенту просматривать ассортимент продуктов и услуг фирмы, просматривать фотографии или изображения продуктов, а также информацию о технических характеристиках продукта и ценах. Интернет-магазины обычно позволяют покупателям использовать функции «поиска», чтобы найти конкретные модели, бренды или предметы. Задача была реализована при использовании шаблона MVC, СУБД MySQL, языка разметки html, языка описания внешнего вида CSS и языка сценариев JavaScript. Также проект был занесён на онлайн-репозиторий GitHub. Пользователям больше не нужно ходить в салоны сотовой связи, чтобы купить себе

телефон. Им помогут настоящие профессионалы выбрать и произвести покупку оригинального и современного товара.

Список использованных источников

Приложения