# 2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Изучив архитектуру веб-приложений и выработав требования необходимые для разработки системы, разрабатываемую систему можно разбить на функциональные блоки (модули). Благодаря этому обеспечивается гибкая архитектура системы. Блоки можно изменять или заменять без изменения других частей системы.

Разрабатываемое приложение состоит из клиентской и серверной части каждую из них можно разбить на несколько блоков.

Клиентская часть состоит из следующих блоков:

– блок логики клиентской части приложения;
– блок пользовательского интерфейса;
– блок фонового обмена данными.

В серверной части можно выделить следующие блоки:

– блок обработки запросов клиента;
– блок валидации запросов;
– блок сервис;
– блок работы с СУБД;
– блок моделей;
– блок обработки ошибок.

Структурная схема, иллюстрирующая названные выше блоки и взаимосвязь между ними, приведена в чертеже ГУИР.400201.089 С1.

Каждый модуль отвечает за данную ему задачу, данные между различными модулями передаются различными форматами, например протоколами.

Рассмотрим функциональные блоки приложения.

## 2.1 Обзор функциональных блоков разрабатываемого приложения

### 2.1.1 Клиентская часть приложения

*Блок логики клиентской части приложения* обеспечивает построение динамичных запросов, в зависимости от введенной пользователем информации. Для этого используются различные инструменты HTML, например тег form. Тэг устанавливает форму на веб-странице. Для ввода текста (данных) используются HTML-теги для ввода данных (текстовое поле, выпадающий список, календарь и т.п.). Для отправления данных на сервер используется теги для подтверждения форм, например кнопки. При конфигурации форм указываются атрибуты направления отправки запроса и HTTP-метод (GET, POST, PUT, DELETE). Большинство запросов пользователя создаются с помощью тега form.

Для перемещения по страницам без каких-либо пользовательских данных, используются ссылки (атрибут href), где указывается нужный адрес

(абсолютный или относительный).

*Блок фонового обмена данными* обеспечивает обмен данными между браузером и веб-сервером, при этом веб-страница не перегружается, а лишь частично обновляется. Это позволяет сделать приложение быстрее и удобнее.

Блок реализован с помощью технологии AJAX. Обмениваться данными можно HTTP запросами (GET и POST). AJAX позволяет управлять содержимым запроса, например устанавливать заголовки и тело. При получении ответа, он обрабатывается и нужные элементы веб-страницы обновляются в соответствии с новыми данными. Благодаря AJAX, пользователь может удобно добавлять продукты в корзину, получить уведомление об результате и продолжать выбирать товары. Для наглядности, ниже принцип работы AJAX и его отличие от метода классических приложений изображен на рисунке 2.1.

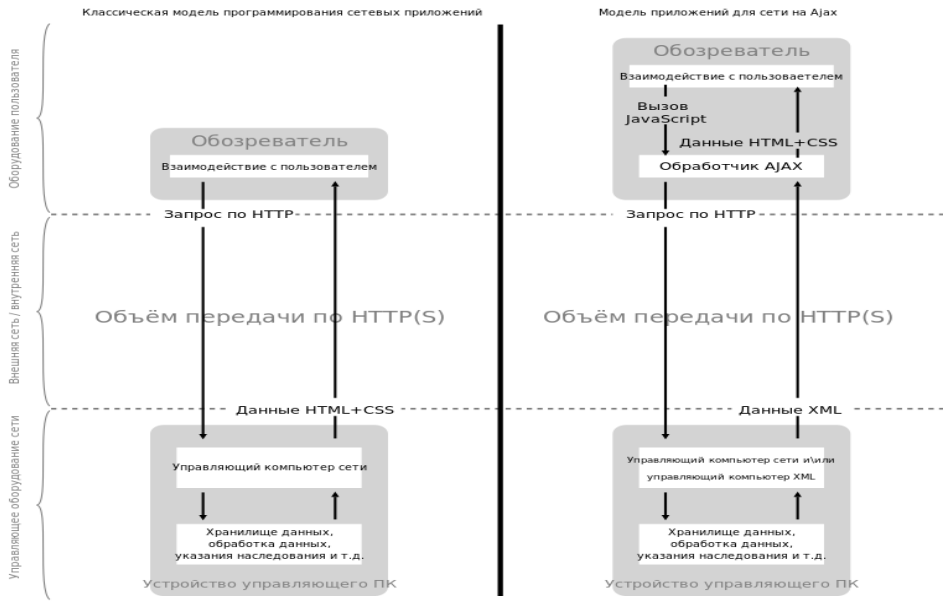


Рисунок 2.1 – Пример работы AJAX.

*Блок пользовательского интерфейса* отвечает за показ данных пользователю в окне веб-браузера и взаимодействие с приложением.

При построении пользовательского интерфейса применяются различные элементы HTML, такие как таблицы, шрифты, кнопки и прочее. Чтобы придать этим элементам красивый вид, правильно расположить их на веб-странице, задать различные характеристики (размер, цвет) используется CSS. Для облегчения разработки пользовательского интерфейса используется набор инструментов Bootstrap (создан на HTML и CSS). Он содержит готовые шаблоны различных элементов веб-страниц. Bootstrap позволяет сделать графическое отображение адаптивным, то есть подходящим под любое разрешение экрана.

В реализуемой системе поддерживаются 2 языка: русский и английский. Для оптимизации количества кода и удобного добавления поддерживаемых языков используется тэг Spring Framework – message. Значения достаются из файлов по ключу, язык устанавливается в зависимости от выбора пользователя.

### 2.1.2 Серверная часть приложения

*Блок обработки запросов клиента* отвечает за приём запросов пользователей, передачу данных другим модулям, и передачу ответов клиентской части. Работа с другими серверными модулями организована так, чтобы они оставались максимально переносимыми. После получения запроса от клиента, выполнение передаётся в блок валидации запросов. После работы других частей приложения формируется ответ. Для наглядности на рисунке 2.2 показан процесс обработки запроса в системе.

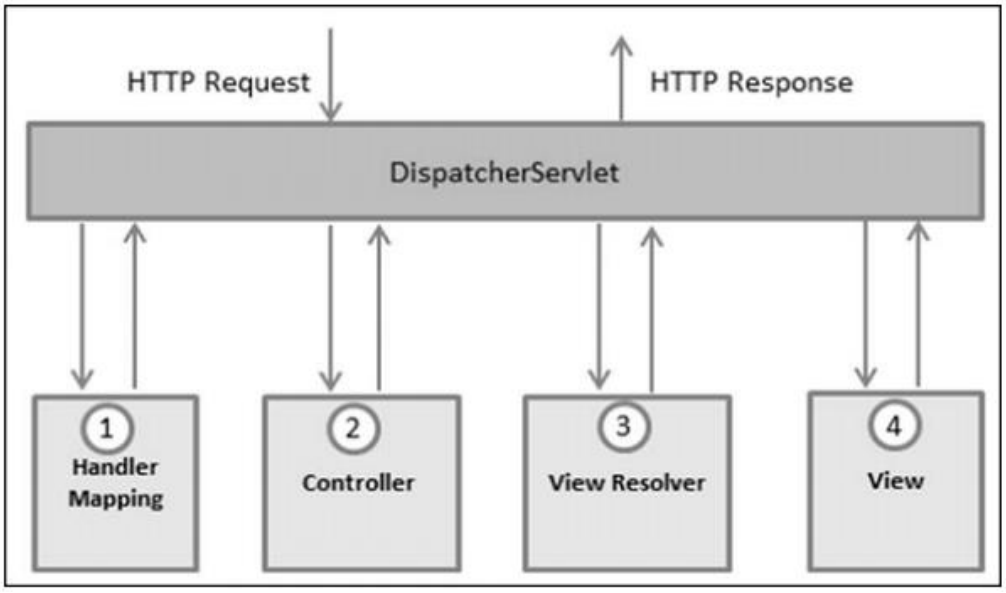


Рисунок 2.2 – Пример обработки запросов в реализуемой системе

При разработке приложения будет использован Spring MVC. Для построения блока используются различные аннотации фреймворка. Например, чтобы определить класс, как контроллер Spring MVC используется аннотация `@Controller`. Аннотация `@RequestMapping` для привязки контроллера к определённому URL и типу HTTP-запроса.

*Блок валидации запросов* содержит в себе логику по проверке запросов пользователей. В блок поступают DTO-объекты (Data Transfer Object).

Обязанности данного блока:

- проверка значений формы на пустоту;
- проверка значений формы на соответствие условиям;

– проверка наличия товаров на складе.

Блок реализован на языке Java с помощью библиотеки Spring Validation. Для проверки значений на правильность используются, как собственные утилиты, так и утилиты `ValidationUtils`, `Regex`. Управление передаётся из блока обработки запросов клиента.

*Блок сервис* предназначен для абстракции логики блока обработки клиентов от блока моделей и блока работы с СУБД. Благодаря реализации блока, можно свободно заменять различные инструменты работы с БД. Так же в блоке будет реализована логика работы с данными, хранящимися в HTTP-сессии.

*Блок работы с СУБД* представляет собой набор классов для обработки SQL запросов к базе данных. Этот блок позволяет сделать приложение независимым от той или иной СУБД, предоставляя классам, использующим его, интерфейс взаимодействия с возможными различными реализациями.

Блок построен с использованием библиотеки Spring JDBC и драйвера для работы с MySQL. Spring Data JDBC – мощный механизм для подключения к базе данных и выполнения SQL запросов. Механизм представляет множество преимуществ, в сравнении с стандартным JDBC API, например:

– уменьшает количество кода, связанное с подключением к БД;
– облегчает обработку исключительных ситуаций;
– не требуется обработка транзакций;
– простой перенос с одного типа СУБД на другую.

Для получения данных из БД используются стандартные SQL-запросы. Полученные данные упаковываются в объекты классов сущностей и возвращаются в блок сервис.

*Блок моделей* является набором классов-сущностей, используемых в работе программы при передаче между различными классами, в ходе работы с базой данных. В данном блоке представлены описания предметов из реальной жизни (в данном случае устройства и характеристики). Набор полей классов позволяет детально описать предметы, для дальнейшего использования в программе.

*Блок обработки ошибок* служит для обработки ошибок, возникших при работе программы (например, страница не найдена, внутренняя проблема сервера). Блок позволяет обрабатывать исключения глобально и вернуть пользователю соответствующую страницу с информацией.

Блок написан с помощью инструментов Spring и Java EE.

## 2.2 Обоснование выбора технологий

### 2.2.1 Клиентская часть

При разработке клиентской части приложения требуется разработать логику, графический интерфейс, реализовать фоновый обмен данными.

Для разработки интерфейса были выбраны языки HTML и CSS, так как они предоставляют множество инструментов для создания веб-страниц, хорошо задокументированы, поддерживаются в практически любом браузере.

Фоновый обмен данными будет реализован с помощью языка JavaScript и технологии AJAX, позволяющей асинхронно коммуницировать с сервером. Инструменты позволяют ускорить работу приложения, а также удобны в использовании.

### 2.2.2 Серверная часть

Разработку серверной части можно условно разделить на задачи, каждая из которых будет реализована с помощью различных инструментов:

– получение информации из базы данных об продаваемых продуктах и и их характеристиках (размеры, цвет и т.п.). Сервис будет реализован с помощью фреймворка Spring (Spring Data JDBC), облегчающим работу с БД;

– бизнес-логика приложения будет реализована с помощью языка программирования Java, в виду его кроссплатформенности, быстродействия;

– часть, отвечающая за обработку запросов пользователей, будет реализована с помощью Spring (Spring MVC), так как этот фреймворк предоставляет мощные инструменты для работы с HTTP;

– модуль, отвечающий за валидацию запросов пользователей, будет реализован с помощью инструментов Spring и Java;

Выбранные технологии и фреймворки обеспечивают быструю разработку приложений, ускоряют развёртывание системы, и хорошо оптимизированы, что увеличивает надежность системы и уменьшает шанс ошибок при разработке.

### 2.2.3 База данных

База данных в системе должна быть быстрой и отказоустойчивой. Операции должны занимать минимальное количество времени. При учёте этих требований была выбрана реляционная база данных MySQL, подходящая для небольших проектов. У неё хорошая система управления, удобная и быстрая развёртываемость. К тому же MySQL распространяется бесплатно.