**ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ**

**3. 1. Разработка структуры приложения**

Основным источником информации для веб-приложения является содержимое базы данных.

Администратор интернет-магазина имеет личный кабинет, который позволяет ему самостоятельно добавлять, редактировать и удалять информацию на сайте при помощи специальных форм.

Формы, представленные на рисунках 3.1-3.3, показывают, как администратор может быстро добавлять новые фитнес-программы, редактировать уже имеющиеся или удалять их. Удобный и понятный интерфейс интернет-магазина позволяет администратору обновлять информацию на сайте без помощи разработчиков и без использования сторонних сервисов.

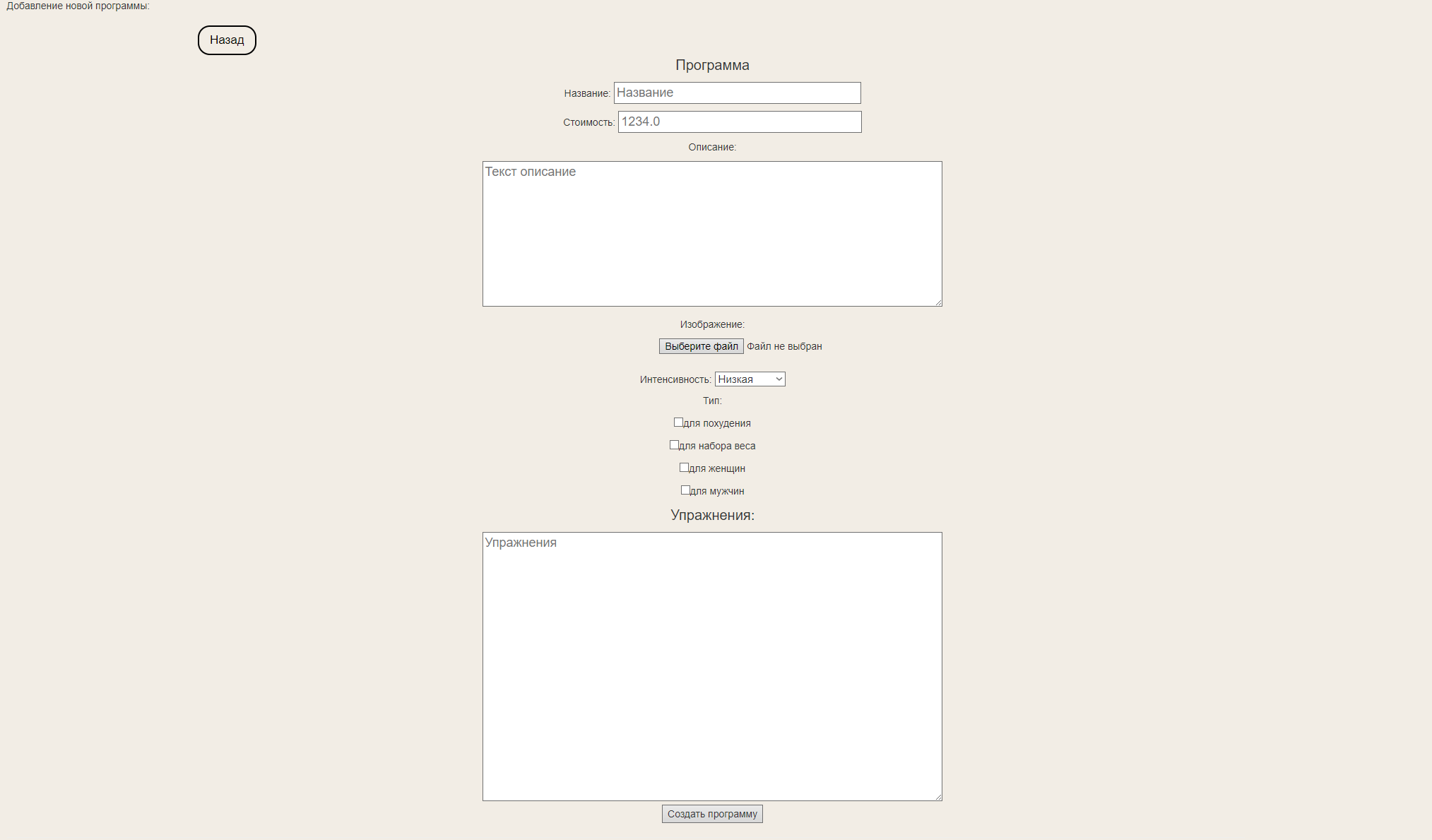


Рисунок 3.1 – Форма для добавления новой фитнес-программы

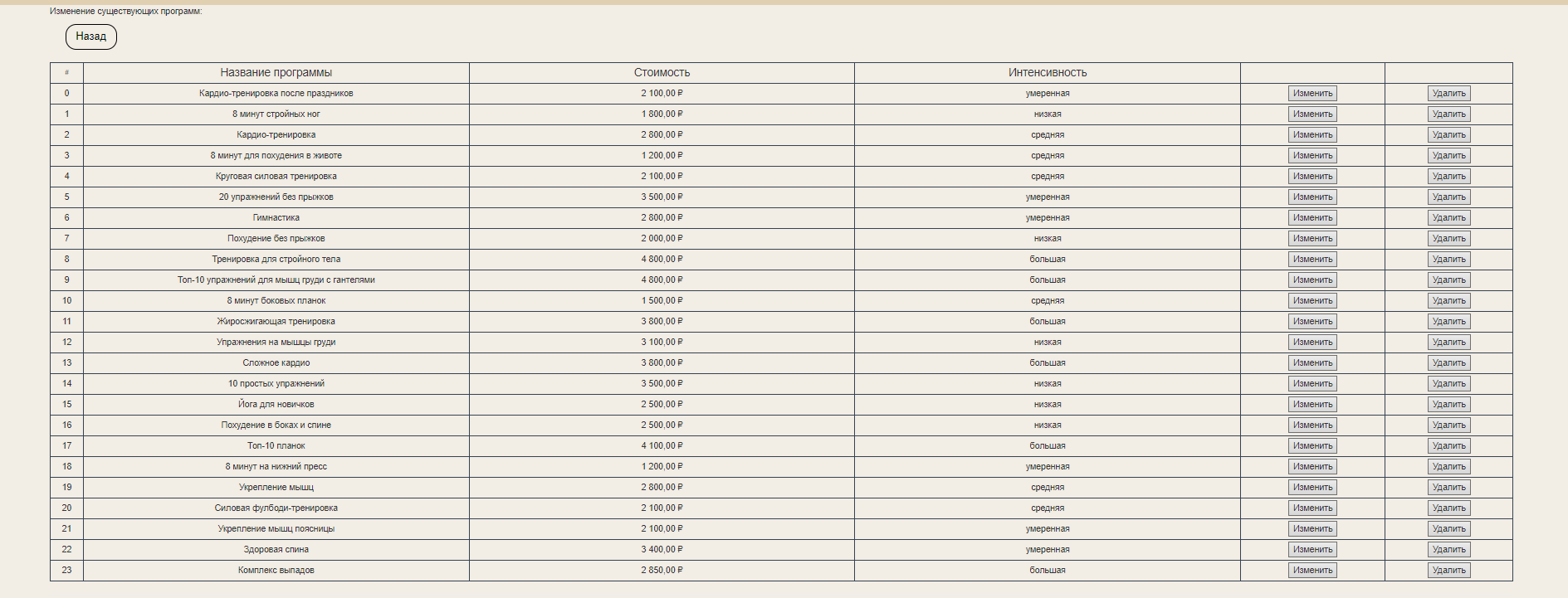


Рисунок 3.2 – Форма для редактирования и удаления фитнес-программы

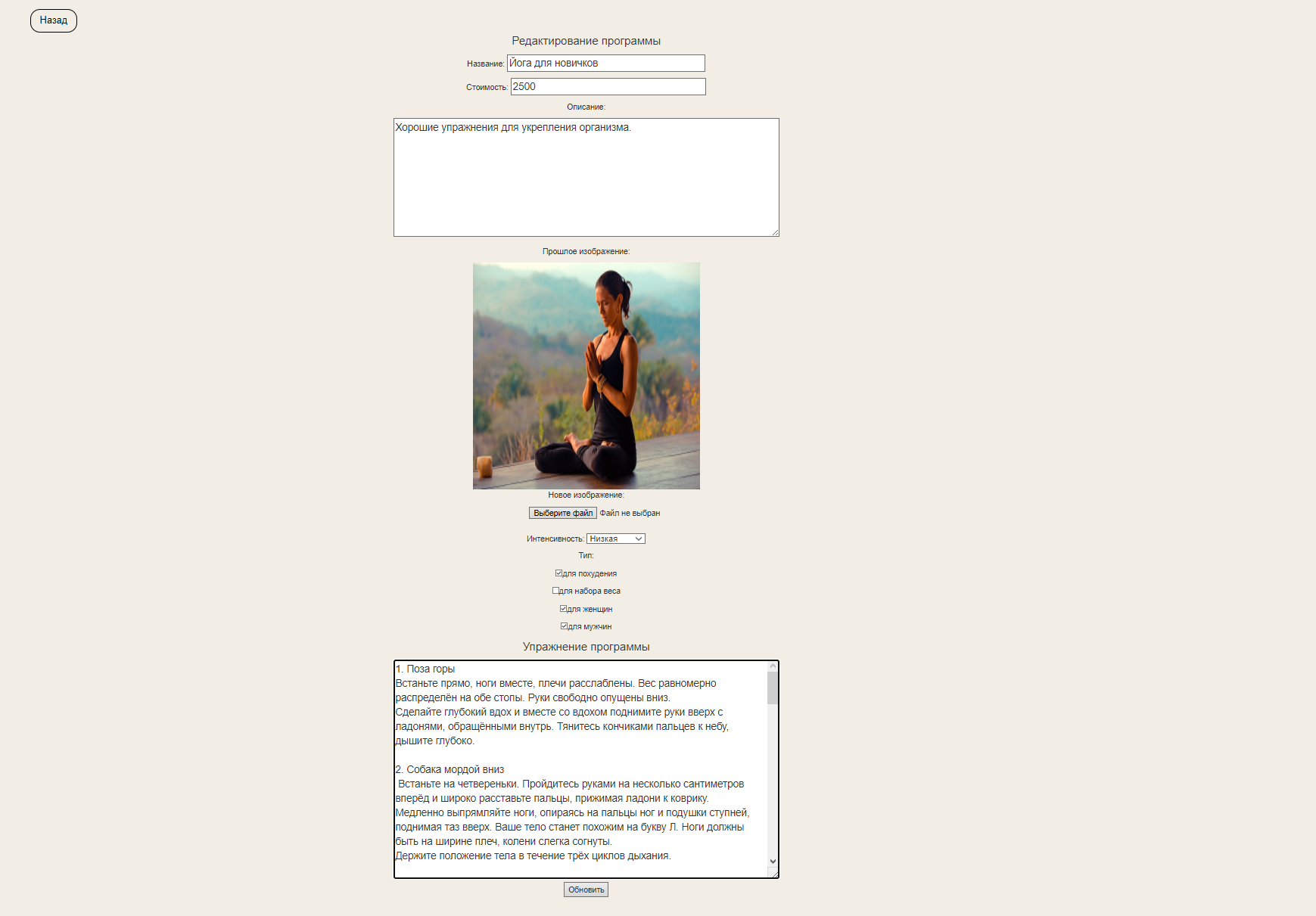


Рисунок 3.3 – Форма для редактирования фитнес-программы

Формы, представленные на рисунках 3.4-3.6, показывают, как администратор может быстро добавлять новое спортивное питание, редактировать уже имеющееся или удалять его. Интерфейс сайта позволяет ему обновлять информацию без помощи разработчиков и без использования сторонних сервисов.

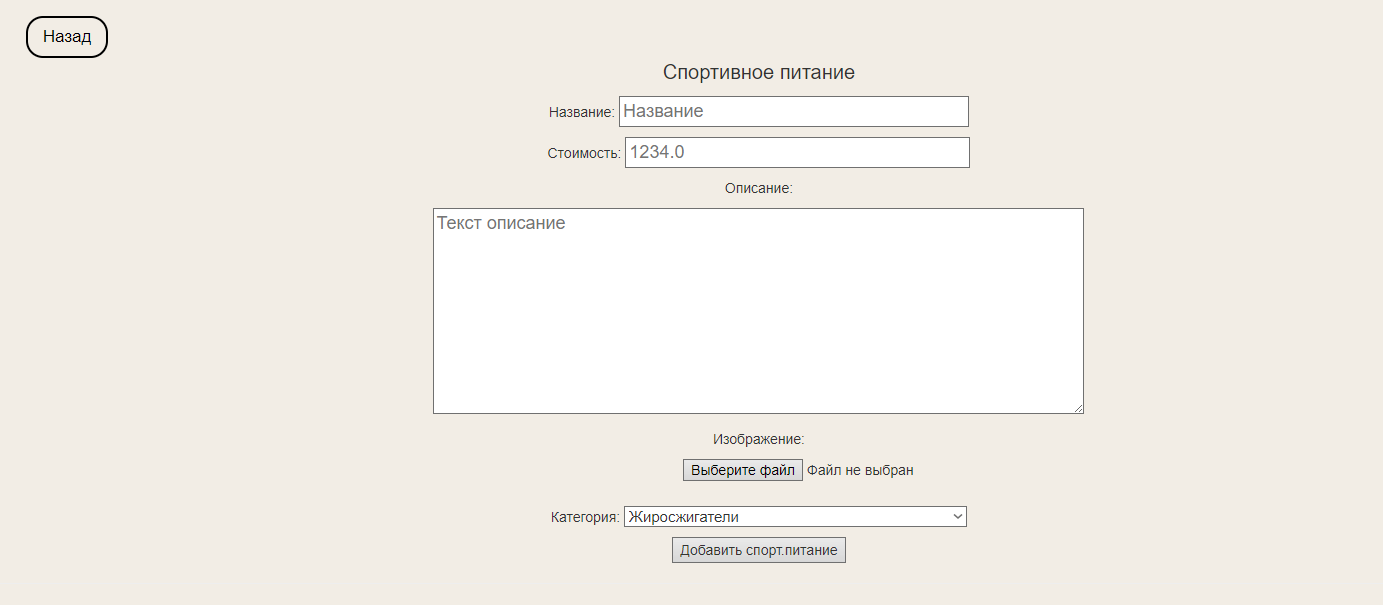


Рисунок 3.4 – Форма для добавления спортивного питания

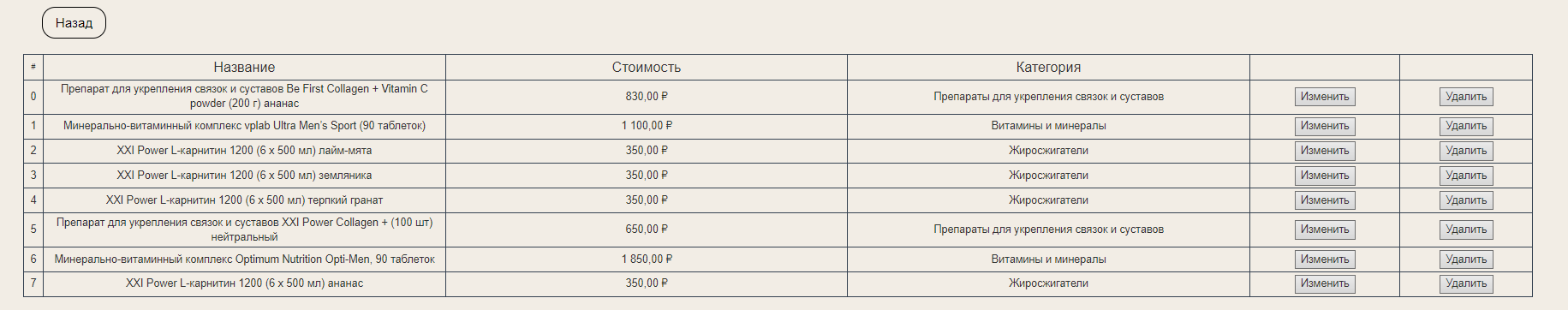


Рисунок 3.5 – Форма для удаления спортивного питания

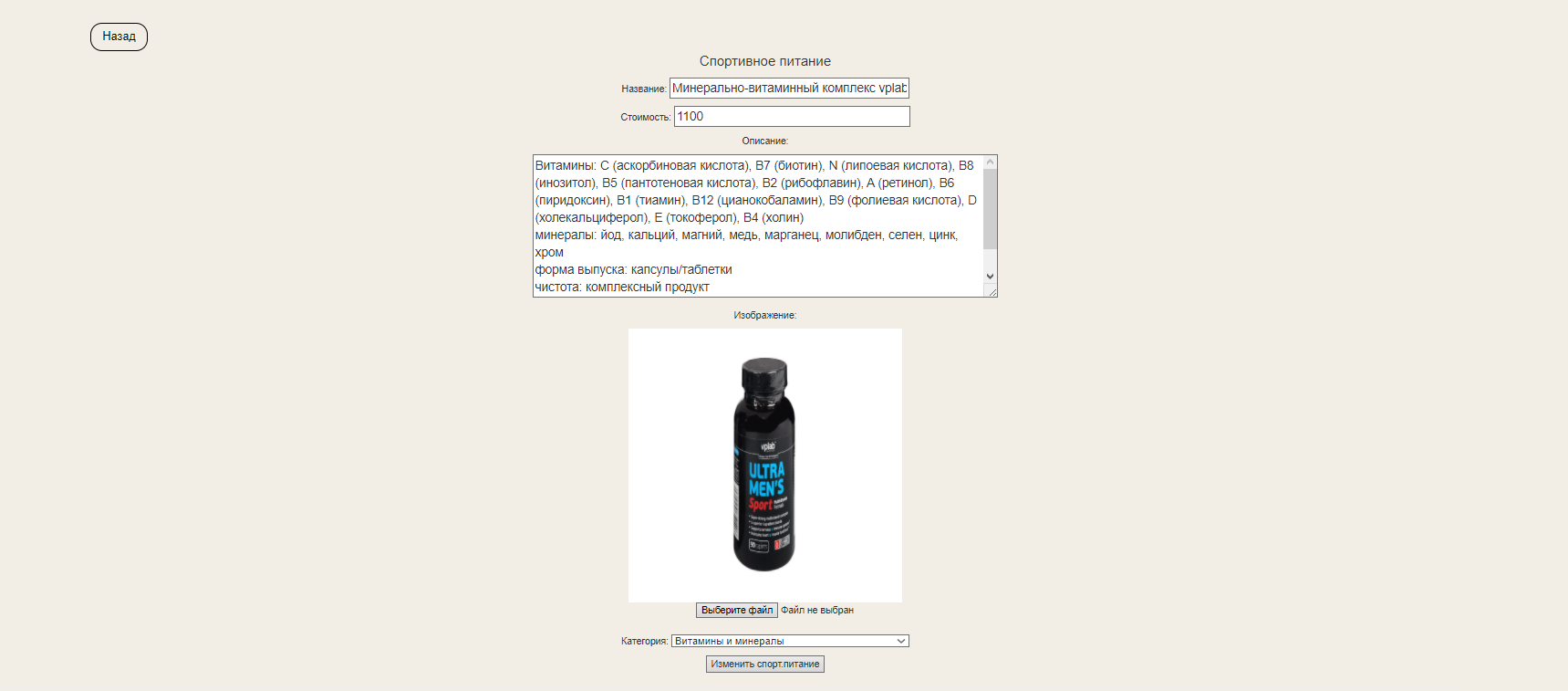


Рисунок 3.6 – Форма для редактирования спортивного питания

При выборе интенсивности или типа фитнес-программы появляется выпадающий список, который дает возможность выбрать интенсивность из четырех вариантов: «низкая», «умеренная», «средняя», «большая» (рисунок 3.7), а также выбрать тип упражнения из четырех вариантов: «для похудения», «для набора веса», «для женщин», «для мужчин» (рисунок 3.8). В разделе «Спортивное питание» предоставляется выбор категории из трех вариантов: «жиросжигатели», «витамины и минералы», «препараты для укрепления связок и суставов» (рисунок 3.9). В зависимости от выбора в базу данных отправляется определенное значение.

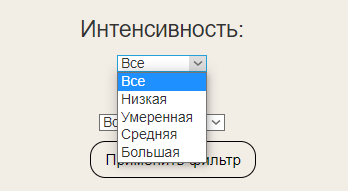


Рисунок 3.7 – Возможность выбора интенсивности программы

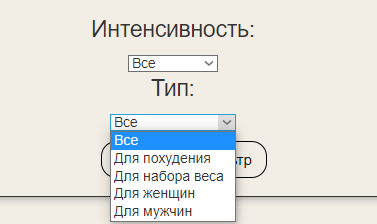


Рисунок 3.8 – Возможность выбора типа программы

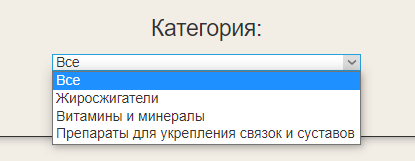


Рисунок 3.9 – Возможность выбора категории питания

Для хранения информации была выбрана система управления базой данных «PostgreSQL». PostgreSQL – это свободная объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом. Она поддерживает большую часть стандарта SQL и предлагает множество функций. Данная система отличается скоростью, оптимизацией для работы с большими массивами информации, а также безопасностью.

На рисунке 3.10 приведен пример запроса для изменения администратором главной страницы интернет-магазина, содержащей четыре фитнес-программы разной степени интенсивности.

* UPDATE – оператор языка SQL, позволяющий обновить значения в заданных столбцах таблицы;
* SET – ключевое слово языка SQL, после которого должен идти список полей таблицы, которые будут обновлены, а также сами новые значения в виде имя поля="значение";
* WHERE – ключевое слово, которое задает условия, ограничивающие обновляемые строки.

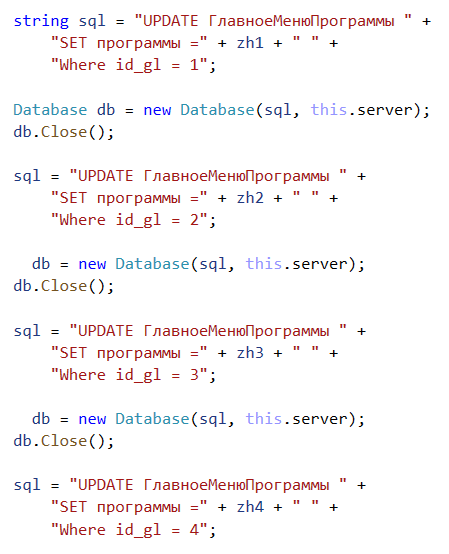


Рисунок 3.10 – Пример запроса для обновления главной страницы

На рисунках 3.11-3.12 представлены запросы, позволяющие зарегистрированному пользователю добавить в корзину фитнес-программы или спортивное питание.

* INSERT – оператор языка SQL, который позволяет добавить строки в таблицу, заполняя их значениями;
* INTO – ключевое слово, которое вставляет в таблицу результирующие строки из запроса;
* VALUES – ключевое слово языка, с помощью которого значения можно вставлять перечислением в круглых скобках через запятую.



Рисунок 3.11 – Пример запроса добавления в корзину программы



Рисунок 3.12 – Пример запроса добавления в корзину спортивного питания

**3. 2. Логическая схема базы данных**

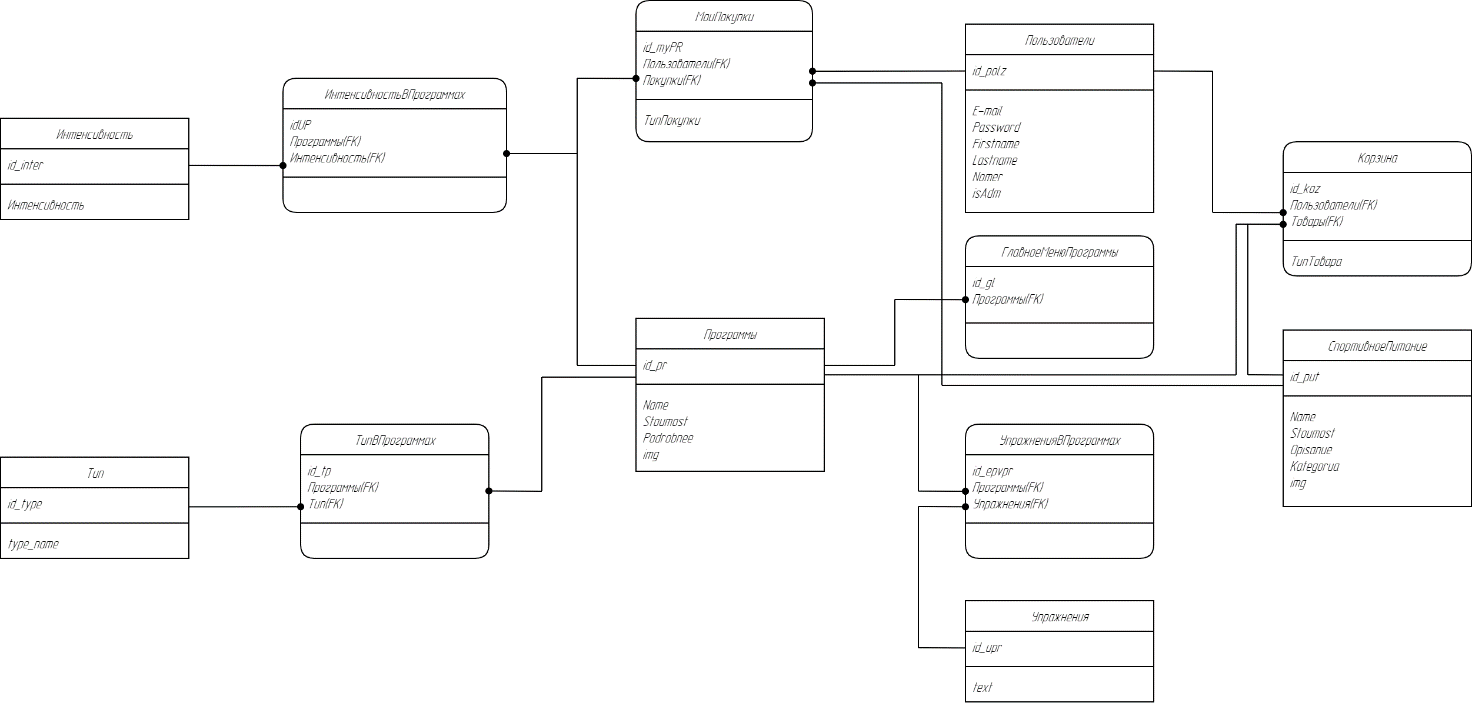


Рисунок 3.13 – Логическая схема

На рисунке 3.13 изображена логическая схема базы данных. Таблица «Пользователи» содержит следующую информацию о каждом пользователе: его идентификатор, адрес электронной почты, пароль от учетной записи, имя, фамилию, номер телефона и роль пользователя в системе. Роль определяет, является ли пользователь администратором интернет-магазина (значение «true» – является, значение «false» – не является).

Каждый зарегистрированный в системе пользователь имеет доступ к таблицам «Корзина» и «МоиПокупки». Таблица «Корзина» хранит следующую информацию: какой конкретно пользователь добавил тот или иной товар, какой именно товар был добавлен и тип данного товара (фитнес-программа или спортивное питание). В таблице «МоиПокупки» сохраняется информация о пользователе и о том, какую именно покупку он совершил (фитнес-программа или спортивное питание).

В таблицах «МоиПокупки» и «Корзина» поле «Пользователи(FK)» использует внешний ключ (foreign key), этот ключ представляет собой общий компонент, связывающий вместе две таблицы. Внешний ключ всегда ссылается на первичный ключ.

Исходная таблица называется родительской, а ссылочная таблица с внешним ключом называется дочерней таблицей. Ссылки на внешние ключи хранятся в дочерней таблице и связаны с первичным ключом в отдельной таблице. Столбец, действующий как внешний ключ, должен иметь соответствующее значение в связанной таблице. Это создает ссылочную целостность. В данном случае родительской таблицей является таблица «Пользователи».

В каждой таблице обязательно должен присутствовать первичный ключ (primary key), этот ключ является одним из потенциальных ключей отношения, который выбирается в качестве основного ключа. Первичный ключ представляет собой поле, которое используется для обеспечения уникальности данных в таблице. В данном случае в любой таблице первичным ключом будет идентификатор (id).

Таблица «ГлавноеМенюПрограммы» содержит идентификатор и информацию о фитнес-программах, которые ссылаются на таблицу «Программы». В этой таблице хранится информация о названии спортивной программы, ее стоимости, кратком описании содержащихся в ней упражнений и фотографии.

В таблице «УпражненияВПрограммах» содержится информация о фитнес-программах, а также упражнениях, которые, в свою очередь, ссылаются на таблицу «Упражнения». Данная таблица хранит информацию об идентификаторе каждого упражнения и его подробное описание (текст).

Таблица «СпортивноеПитание» содержит идентификатор каждого продукта, его название, стоимость, краткое описание, категорию, к которой он относится, и фотографию.

Таблица «ИнтенсивностьВПрограммах» содержит поля, ссылающиеся на таблицы «Программы» и «Интенсивность». Последняя хранит информацию о каждом типе интенсивности фитнес-программы и его идентификаторе.

Таблица «ТипВПрограммах» имеет поля, которые ссылаются на таблицу «Программы» и таблицу «Тип». Последняя хранит информацию о каждом типе фитнес-программы и его идентификаторе.

**3.3. Разработка интерфейса взаимодействия пользователя с системой**

Перед созданием интерфейса был разработан прототип, в котором было выбрано расположение кнопок и расположение меню. Прототип представлен на рисунке 3.14.

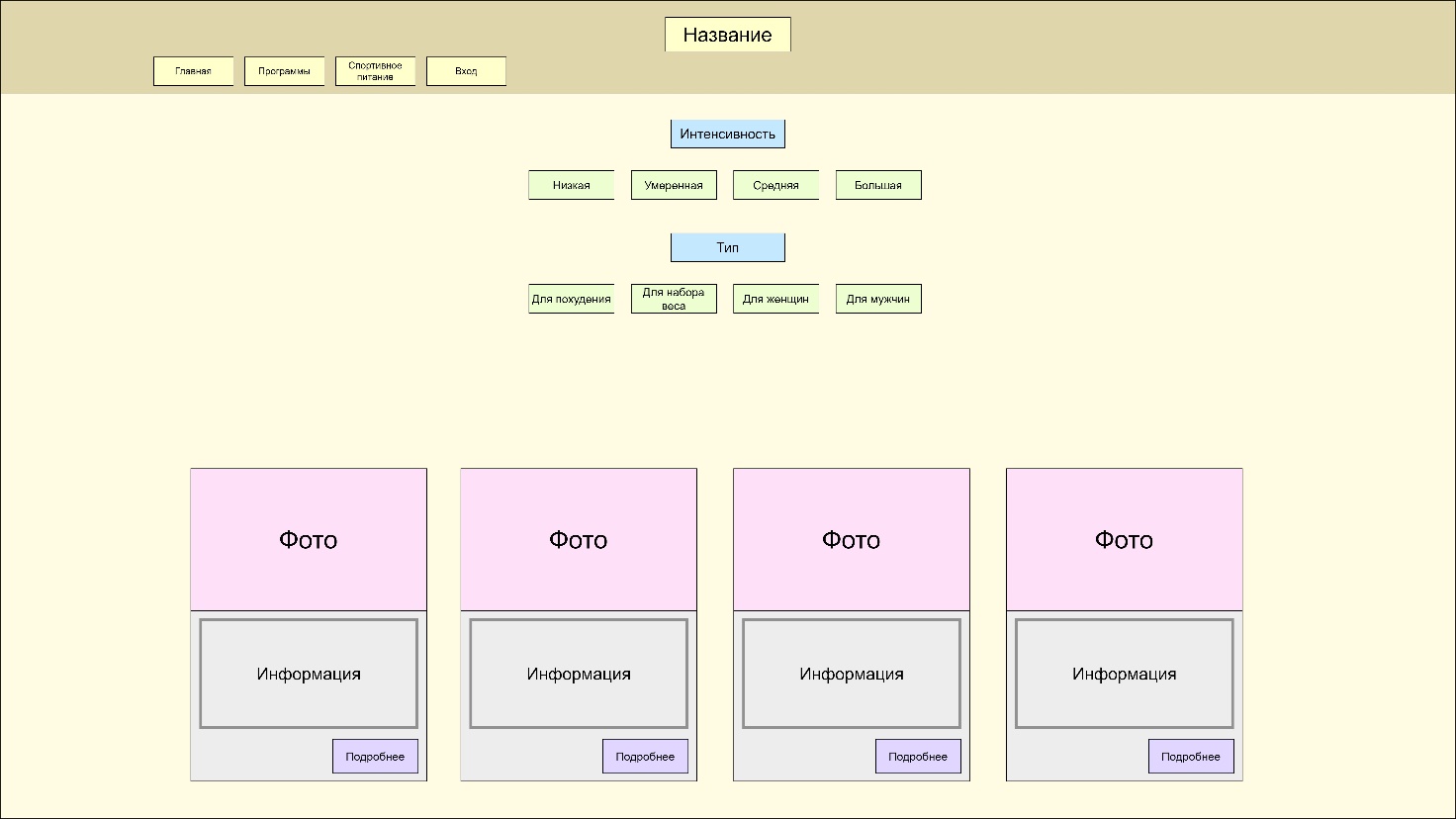


Рисунок 3.14 – Прототип интерфейса веб-приложения

После создания прототипа был разработан интерфейс интернет-магазина фитнес-программ для удобного взаимодействия с пользователем (рисунок 3.15).

Пользователю, являющемуся гостем, предоставляется возможность просмотра главной страницы сайта, выбора всех программ по интенсивности и типу, выбора спортивного питания по типу, а также возможность регистрации, если он не сделал этого ранее, и авторизации, если пользователь уже зарегистрирован. Формы для регистрации и авторизации представлены на рисунках 3.16-3.17.

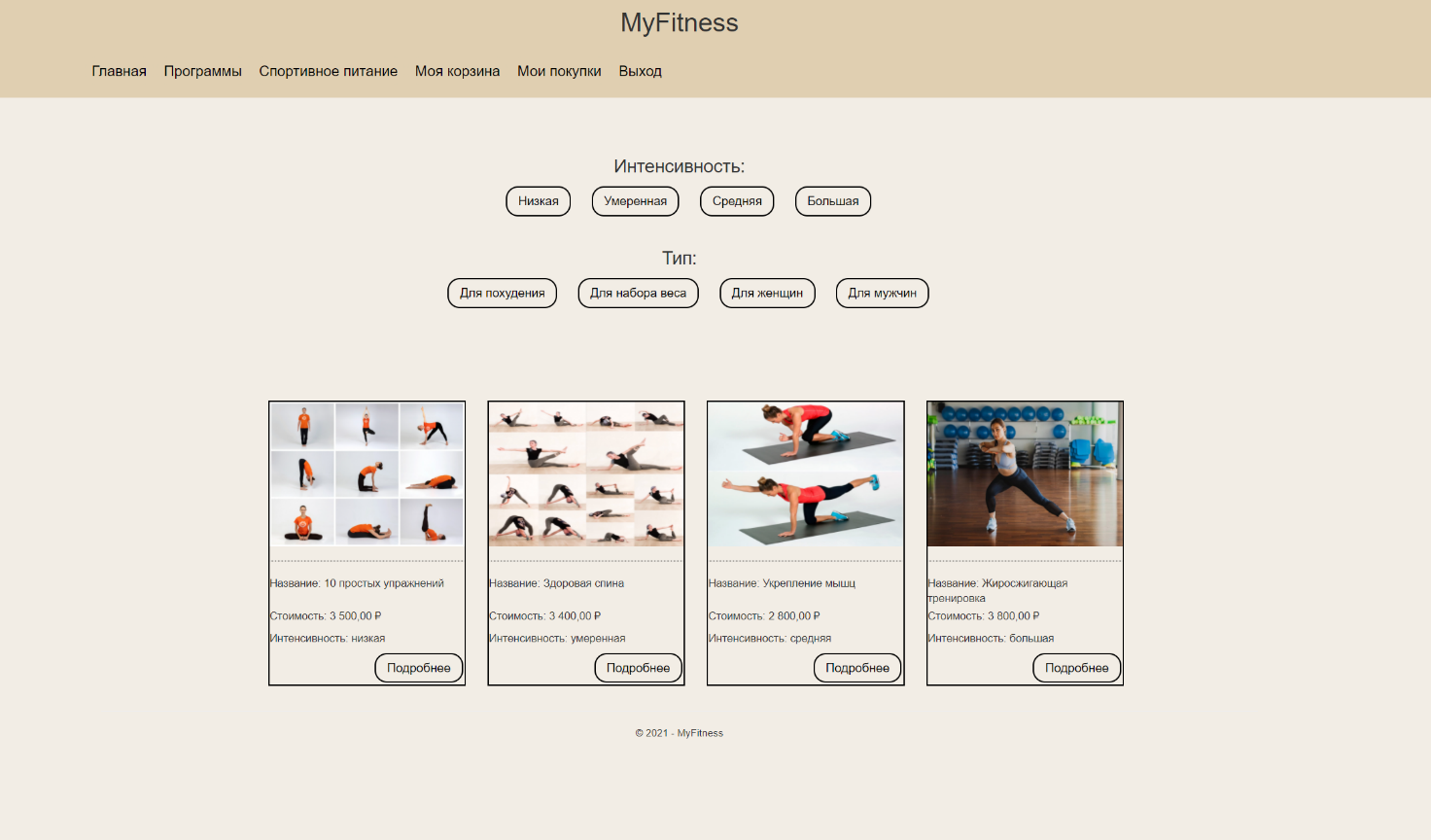


Рисунок 3.15 – Главная страница интернет-магазина «MyFitness»

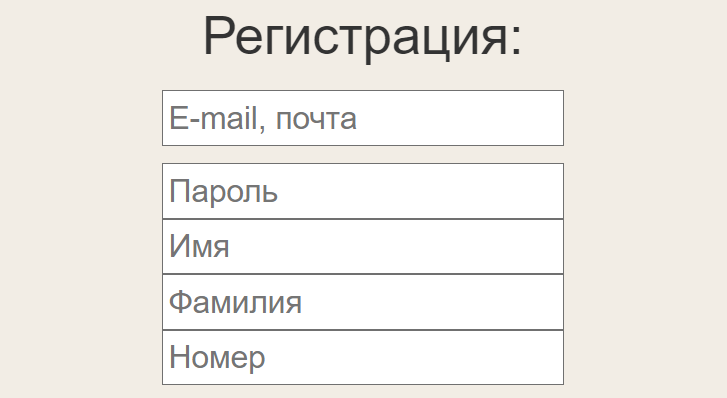


Рисунок 3.16 – Страница «Регистрация»

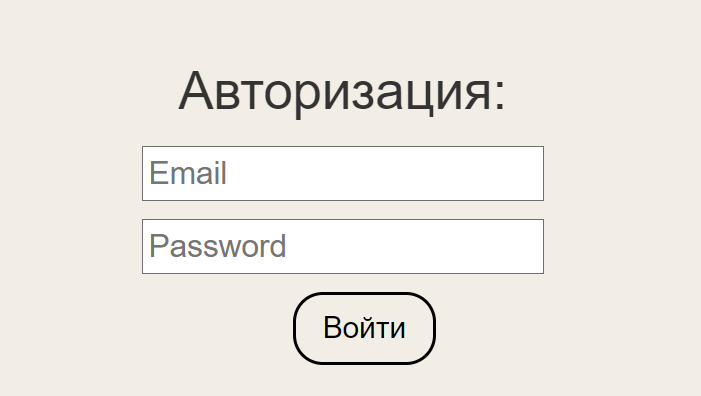


Рисунок 3.17 – Страница «Авторизация»

При введении некорректных данных пользователю выводится сообщение об ошибке (рисунок 3.18). Это может быть ввод неправильного пароля или email-адреса.

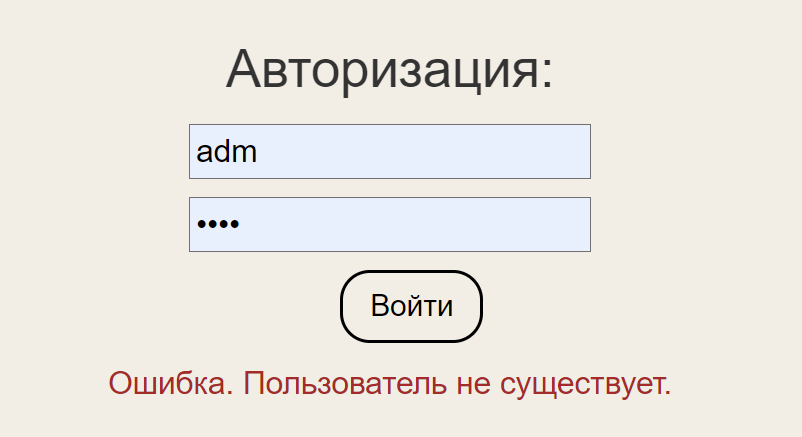


Рисунок 3.18 – Сообщение об ошибке

Пользователю, являющемуся администратором, предоставляется возможность добавления, редактирования и удаления информации на сайте (рисунки 3.19-3.20).

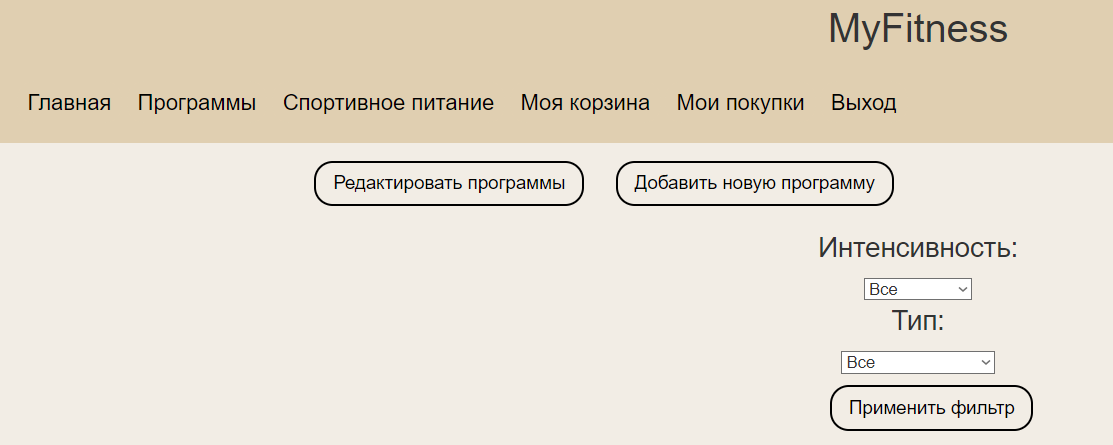


Рисунок 3.19 – Появление возможности добавления и редактирования программ

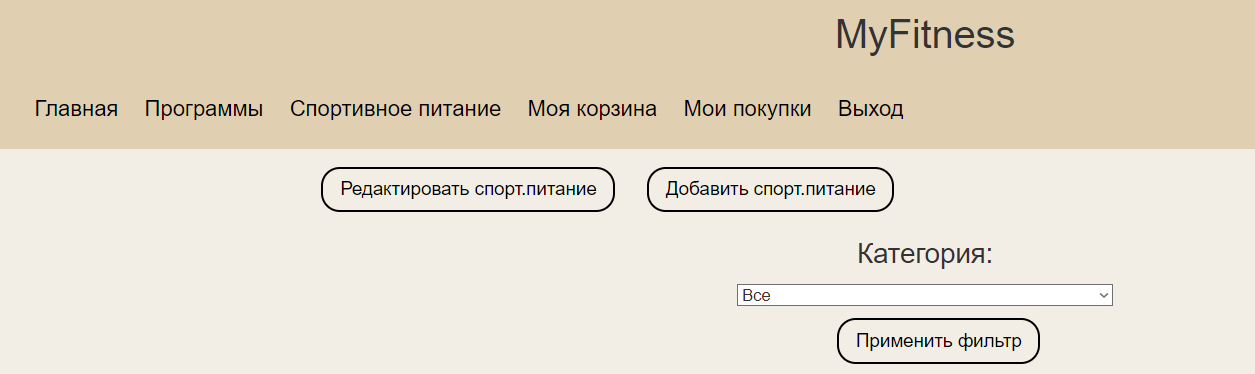


Рисунок 3.20 – Появление возможности добавления и редактирования спортивного питания

После регистрации или авторизации пользователю доступна функция совершения покупки и просмотра всех своих заказов, которые он сделал за все время использования приложения. Для покупки фитнес-программы или спортивного питания необходимо добавить товар в корзину, нажав на кнопку с соответствующим названием. На странице «Моя корзина» отображается информация о покупке, которую пользователь хочет приобрести (рисунок 3.21). Также пользователь может просматривать все товары в корзине, удалять их из нее или делать заказ. После нажатия кнопки «Оплатить» заказ появится в разделе «Мои покупки» (рисунок 3.22).



Рисунок 3.21 – Страница «Моя корзина»

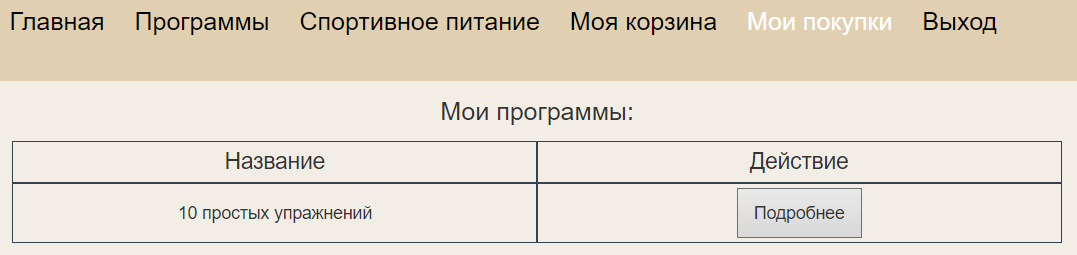


Рисунок 3.22 – Страница «Мои покупки»

**3.4. Разработка архитектуры приложения**

В данной курсовой работе для разработки веб-приложения использовался паттерн MVC.

Паттерн MVC («Model-View-Controller» или «модель-вид-контроллер») – это шаблон архитектуры ПО, который подразумевает разделение программы на три слабосвязанных компонента, каждый из которых отвечает за свою сферу деятельности.

Паттерн MVC включает ряд компонентов:

1. **Model** – модель, предоставляющая доступ к данным. Позволяет извлекать данные и менять их состояние;
2. **View** – представление, отображающее данные клиенту. В веб-программировании существует в виде конечных данных, которые получает клиент;
3. **Controller** – контроллер, отслеживающий различные события (действия пользователя) и по заданной логике оповещающий модель о необходимости изменить состояние системы.
4. **Система маршрутизации** – дополнительный компонент, который сопоставляет запросы с маршрутами и выбирает определенный контроллер для обработки запросов.

Ключевая идея MVC состоит в том, что любое приложение с пользовательским интерфейсом в первом приближении можно разбить на два модуля:

1. модуль, отвечающий за реализацию бизнес-логики приложения;
2. пользовательский интерфейс.

В первом модуле будет реализован основной функционал приложения. Данный модуль будет ядром системы, в котором реализуется модель предметной области приложения. В концепции MVC данный модуль будет обозначать модель (Model).

Во втором модуле будет реализован весь пользовательский интерфейс, включая отображение данных пользователю и логику взаимодействия пользователя с приложением.

Основная цель паттерна MVC — отделить реализацию бизнес-логики приложения (модели) от ее визуализации (вида). Такое разделение повысит возможность повторного использования кода.

Польза применения MVC наиболее наглядна в случаях, когда пользователю нужно предоставлять одни и те же данные в разных формах. Например, в виде таблицы, графика или диаграммы (используя различные виды). При этом, не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью по кнопке, ввод данных).

Если следовать паттерну MVC, можно упростить написание программ, повысить читаемость кода, сделать легче расширение и поддержку системы в будущем.

В данном случае приложению приходит запрос, и система маршрутизации является необходимой для выбора конкретного контроллера, который нужен для обработки запроса. После чего контроллер запускает процесс обработки запроса. Пока контроллер обрабатывает запрос, он может обращаться к данным, используя модели (Model), а как визуализацию ответа использовать представления (View).

Обработка контроллера имеет конечный результат, который отправляется в качестве ответа клиенту. Ответ представляет собой html-страницу, которую пользователь видит в своем браузере.

Связь приложения и сервера осуществляется через протокол HTTP и происходит по принципу REST.

HTTP («HyperText Transfer Protocol» или «протокол передачи гипертекста») – протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

* потребителей (клиентов), которые инициируют соединение и посылают запрос;
* поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

REST («Representational State Transfer» или «передача репрезентативного состояния») – это архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

Свойства архитектуры REST:

1. масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов;
2. открытость компонентов к возможным изменениям;
3. прозрачность связей между компонентами системы для сервисных служб;
4. переносимость компонентов системы путем перемещения программного кода вместе с данными;
5. надёжность, выражающаяся в устойчивости к отказам на уровне системы при наличии отказов отдельных компонентов, соединений или данных.

Система REST работает на уровне HTTP и использует следующие методы для осуществления работы с ресурсами на сервере:

* POST – для создания ресурса;
* GET – для получения ресурса;
* PUT – для обновления ресурса;
* DELETE – для удаления ресурса.

В случае интернет-магазина фитнес-программ необходима возможность добавления новых спортивных программ, просмотра всех уже существующих программ на сайте и просмотра всех оформленных заказов, а также добавления нового спортивного питания и просмотра продуктов из данной категории на сайте.