**Министерство науки и высшего образования Российской Федерации**

Калужский филиал федерального государственного бюджетного

образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана***

***(национальный исследовательский университет)» (КФ МГТУ им. Н.Э. Баумана)***

**ФАКУЛЬТЕТ *ИУК «Информатика и управление»***

## КАФЕДРА ИУК5 «Системы обработки информации»

**Р А С Ч Е Т Н О - П О Я С Н И Т Е Л Ь Н А Я**

**З А П И С К А**

**к курсовой работе на тему:**

«Интернет магазин фитнес-программ»

по дисциплине ***Базы данных***

### Студент гр. ИУК5-51Б ( Шилкина М.С.\_\_ )

(подпись) (Ф.И.О.)

### Руководитель ( Кириллов В.Ю. )

(подпись) (Ф.И.О.)

### Оценка руководителя баллов

30-50 (дата)

|  |  |  |
| --- | --- | --- |
| Оценка защиты | баллов  30-50 | (дата) |
| Оценка проекта | баллов | (оценка по пятибалльной шкале) |

Комиссия: ( )

(подпись) (Ф.И.О.)

( )

(подпись) (Ф.И.О.)

( )

(подпись) (Ф.И.О.)

Калуга, 2020

Калужский филиал

федерального государственного бюджетного образовательного учреждения высшего образования

#### «Московский государственный технический университет имени Н.Э. Баумана

***(национальный исследовательский университет)» (КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **ИУК5**

( )

« » 20 г.

**З А Д А Н И Е**

# на выполнение курсовой работы

по дисциплине ***Базы данных***

Студент *Шилкина М.С., ИУК5-51Б*

(фамилия, инициалы, индекс группы)

Руководитель

*Кириллов В.Ю.*

(фамилия, инициалы)

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 10 нед., 100% к 14 нед.

#### Тема курсовой работы

1. ***Техническое задание***

***«Интернет магазин фитнес-программ»***

*Выполнить исследование и описание предметной области, произвести анализ*

*объекта автоматизации, разработать методы и способы решения технических*

*задач. Разработать и реализовать структуру базы данных, серверную компоненту,*

*интерфейс приложения и взаимодействие с базой данных.*

#### Оформление курсовой работы

* 1. Расчетно-пояснительная записка на листах формата А4.
  2. Перечень графического материала КР (плакаты, схемы, чертежи и т.п.)

Дата выдачи задания « » 20 г.

Руководитель курсовой работы / /

(подпись) (Ф.И.О.)

Задание получил / / « » 20 г.

(подпись) (Ф.И.О.)

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**ОГЛАВЛЕНИЕ**

[1.ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc58808929)

[2.Научно-исследовательская часть 6](#_Toc58808930)

[2.1 Постановка задачи проетирования 6](#_Toc58808931)

[2.2 Описание предметной области 6](#_Toc58808932)

[2.3 Анализ аналогов и прототипов 8](#_Toc58808933)

[2.4.Перечень задач, подлежащих решению в процессе разработки. 9](#_Toc58808934)

[2.5 Обоснование выбора инструментов и платформы для разработки. 10](#_Toc58808935)

[3. Проектно-конструкторская часть 11](#_Toc58808936)

[3.1. Разработка структуры приложения 11](#_Toc58808937)

[3.2. Логическая схема базы данных 17](#_Toc58808938)

[3.3.Разработка интерфейса взаимодействия пользователя с системой 19](#_Toc58808939)

[3.4. Разработка архитектуры приложения 23](#_Toc58808940)

[4. Проектно-технологическая часть 25](#_Toc58808941)

[4.1. Проектирование начального и тестового наполнения базы данных. Порядок развертывания системы. 25](#_Toc58808942)

[Заключение 26](#_Toc58808943)

[Список использованных источников 27](#_Toc58808944)

# 

# 1.ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1. Наименование.

Тема: «Интернет магазин фитнес-программ».

2. Основание для разработки.

* Учебный план специальности **09.03.01** – СОИ
* Рабочая программа дисциплины "Базы данных"

3. Исполнитель.

Студент 3-го курса МГТУ им. Баумана по направлению «Информатика и вычислительная техника» профиль «Системы обработки информации» Шилкина М.С.

4. Цель разработки.

Основная учебная цель выполнения разработки – повышение уровня квалификации разработчика в области проектирования, программной реализации и анализа сложных структур данных и алгоритмов их обработки.

Разрабатываемая система предназначена для покупки фитнес-программ. Со стороны обычного пользователя в ней будет возможность просмотра существующих программ с выбором по интенсивности и типу, и прочей информации, касающейся данной программы. Также пользователь сможет просматривать информацию о его покупках с помощью личного кабинета. Администратор магазина сможет добавлять и изменять информацию на сайте.

5. Содержание работы:

5.1. Задачи, подлежащие решению.

* Исследование и описание предметной области
* Разработка системы, предназначенной для совершения покупок фитнес-программ
* Со стороны обычного пользователя реализовать возможность просмотра существующих программ с выбором по интенсивности и типу, и прочей информации, касающейся данной программы (например, полная информация о программе).
* Со стороны администратора магазина реализовать возможность добавления и изменения информации на сайте.
* Документирование проекта в соответствии с установленными требованиями

5.2. Требования к архитектуре АСОИ.

* простота обслуживания и эксплуатации (минимальные требования к квалификации и дополнительному обучению эксплуатирующего персонала);
* экономичность (экономическая эффективность в процессе функционирования);
* модифицируемость (возможность перенастройки для работы с другими технологическими процессами);
* функциональная расширяемость (возможность ввода в систему дополнительных функциональных возможностей, не предусмотренных в техническом задании);

5.4. Требования к данным.

Система должна быть централизованной, т.е. все данные должны располагаться в базе данных.

5.6. Требования к временным характеристикам.

Требования к временным характеристикам программы не предъявляются.

5.7. Требования к надежности

Устойчивость сервиса при вводе некорректных данных.

5.8. Требования к безопасности

- базовая защита базы данных от выгрузки данных с использованием подмены запроса;

- шифрование пароля пользователя при регистрации в системе.

6. Этапы разработки.

1 этап – изучение предметной области, определение состава и структуры данных. Результат: логическая структура данных, ТЗ, описание объекта автоматизации и целей разработки.

2 этап – анализ пользовательских сценариев работы. Результат: пользовательский интерфейс.

3 этап – анализ взаимодействия веб-приложения с серверной компонентой. Результат: архитектура приложения.

7. Техническая документация, предъявляемая по окончании работы.

* техническое задание по разрабатываемой системе;
* расчётно-пояснительная записка по разработанной системе;
* сопровождающая презентация в формате .ppt или .pptx.

8. Дополнительные условия.

Результаты работ в течение их выполнения загружаются на сервис GitHub, где преподаватель может контролировать процесс выполнения работ. https://github.com/mary-shilkina/train\_code

# 2.Научно-исследовательская часть

# 2.1 Постановка задачи проетирования

Задачей проектирования данной курсовой работы является реализация веб-приложения, предназначеного для покупки фитнес-программ.

# 2.2 Описание предметной области

В современном мире досточно много процессов в разных областях нашей жизни автоматизируется. Автоматизация позволяет облегчить выполнение этих процессов. Например, очень часто мы ходим купить какой-то товар или услугу, не выходя их дома, и интернет-магазины помогают удовлетворить эту потребность, даже если физически магазин находится в другой точке мире, другом городе, или товар (услуга) является нематериальным или цифровым. Таким товаром является фитнес-программа.

Фитнесс является популярной темой в нашем современном обществе. Каждый стремится вести здоровый образ и заниматься спортом.

Очень часто фитнесс-программы продаются в спортивных центрах или, в так называемых, центрах здоровья. Для кого эти центры не доступны, территорально, материально или из-за субъективных причин, а возможность приобрести программу в интернете является отличным решением этой проблемы. Благодаря интренет-магазину клиент сможет быстро, доступно и удобно совершить покупку. Это является большим преимуществом по сравнению с покупкой напрямую в центре.

Интернет-магазин дает возможность клиенту просматреть все возможные виды программ, отдельное описание каждой программы, ее стоимость, позволяя ему выбрать нужную программу искодя из своиx предпочтений, требований и интенсивности. Красивая фотография программы, преемлемая цена и возможность купить «прямо сейчас» замотивирует клиента на покупку.

Клиент может добавить в корзину выбранные программы и оформить заказ. Корзина представляет собой таблицу, содержащую список программ, их количество и общую сумму заказа.

После оформления заказа и оплаты в разделе «Мои программы» будет отображена полная информация о заказе: телефон и e-mail пользователя, стоимость, код заказа, дата заказы и купленные программы.

Незарегистрированный пользователь (гость) имеет возможность только просмотреть главную страницу и информацию о существующих программах без возможности совершения покупки.

Пользователь должен зарегистрироваться на сайте или войти, чтобы совершить покупку. Для этого ему необходимо при регистрации указать имя, логин (в виде email), пароль и телефон. При входе вводится только логин и пароль.

Пользователем может быть администратор (центра или сам предприниматель (например, блогер)). Администратор будет иметь возможность добавления нужной фитнес-программы, ее редактирования полостью или частично. Данная функциональность и удобный интерфейс позволит автоматизировать написание и редактирование вручную.

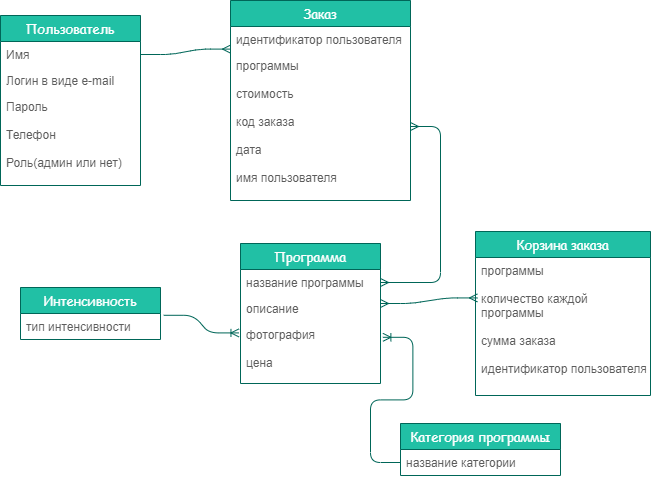


Рисунок 1 — Концептуальная схема

Стоит отметить, что программа должна обязательно относиться к одному из типов «Питание», «Тренировки» или «Гайды». И иметь интенсивность «Легко», «Средне» или «Сложно».

# 2.3 Анализ аналогов и прототипов

Тема здорового образа жизни является актуальной в современном мире, поэтому существует досточное количество аналогов. В большинстве случаев фитнесс-блогеры осуществяют продажу своих фитнес-программ через интернет, но не у всех есть интернет-магазины.

1. <https://my-mission.ru/>

Интернет-магазин фитнес-программ Анастасии Мироновой. Является популярным блогером в России. Интернет-магазин имеет интересный дизайн и предоставляет большой выбор фитнес-программ. Имеется возможность выбора исходя из возраста, интесивности и целей. При нажатии на программу появляется полная информация. Покупка осуществляется пошагово. Также существует личный кабинет, в котором зарегестрированный пользователь после покупки может просматривать купленную программу и информацию о платеже.

1. <https://rebrovass.ru/gotovieprogrammi>

Интернет-магазин фитнес-программ Ребровой Анастасии. Интернет-магазин имеет интересный дизайн и предоставляет небольшой выбор фитнес-программ. Имеется возможность выбора исходя интесивности и целей. Покупка осуществляется с помощью заполнения формы с личной информацией. Личный кабинет отсутвует.

Исследование аналогичных разработок приводит к выводу о том, что, несмотря на большое количество данных разработок, очень немногие предоставляют удобный интерфейс как для пользователя, так и для администратора. Большинство сайтов имеют одностраничную структуру с возможностью совершить покупку через заполненную форму. Это не всегда является удобным для пользователя, так как программа может затеряться со временем или вовсе не прийти на почту. А также пользователь не может убедиться в успешном совершении заказа. Администратор или владелец магазина в свою очередь не имеет возможности без помощи разработчиков оперативно изменять и добавлять информацию о новых программах.

# 2.4.Перечень задач, подлежащих решению в процессе разработки.

Для создания веб-приложения необходимо:

1)Определить состав и структуру данных;

2)Спроектировать пользовательский интерфейс;

4)Определить архитектуру приложения;

5)Написать серверную часть и дополнить клиенскую часть;

6)Обнаружить и исправить возникающие ошибки.

# 2.5 Обоснование выбора инструментов и платформы для разработки.

Для разработки данного веб-приложения будет использоваться платформа Node.

Node.js (или просто Node) — это серверная платформа для работы с [JavaScript](https://netology.ru/programs/javascript) через движок V8. JavaScript выполняет действие на стороне клиента, а Node — на сервере. С помощью Node можно писать полноценные приложения. Node умеет работать с внешними библиотеками, вызывать команды из кода на JavaScript и выполнять роль веб-сервера. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript.

Express - это фреймворк для Node js, который реализовывает слой функций, необходимых для создания эффективных приложений и API. Его использование значительно сокращает написание кода, а, значит, уменьшается затрачиваемое на разработку время.

В Node js для генерации и отдачи HTML-страниц используются шаблонизаторы. Node js шаблонизатор представляет собой специальный модуль, использующий более удобный синтаксис для формирования HTML на основе динамических данных и позволяющий разделять представление от контроллера. В качестве шаблонизатора будем использовать Handlebars.

# 3. Проектно-конструкторская часть

# 3.1. Разработка структуры приложения

Содержимое базы данных является основным источником информации для данного веб-приложения.

С помощью личного кабинета администратор с помощью специальных форм вручную заносит данные о новых программах, изменяет информацию о них и удаляет, в случае необходимости.

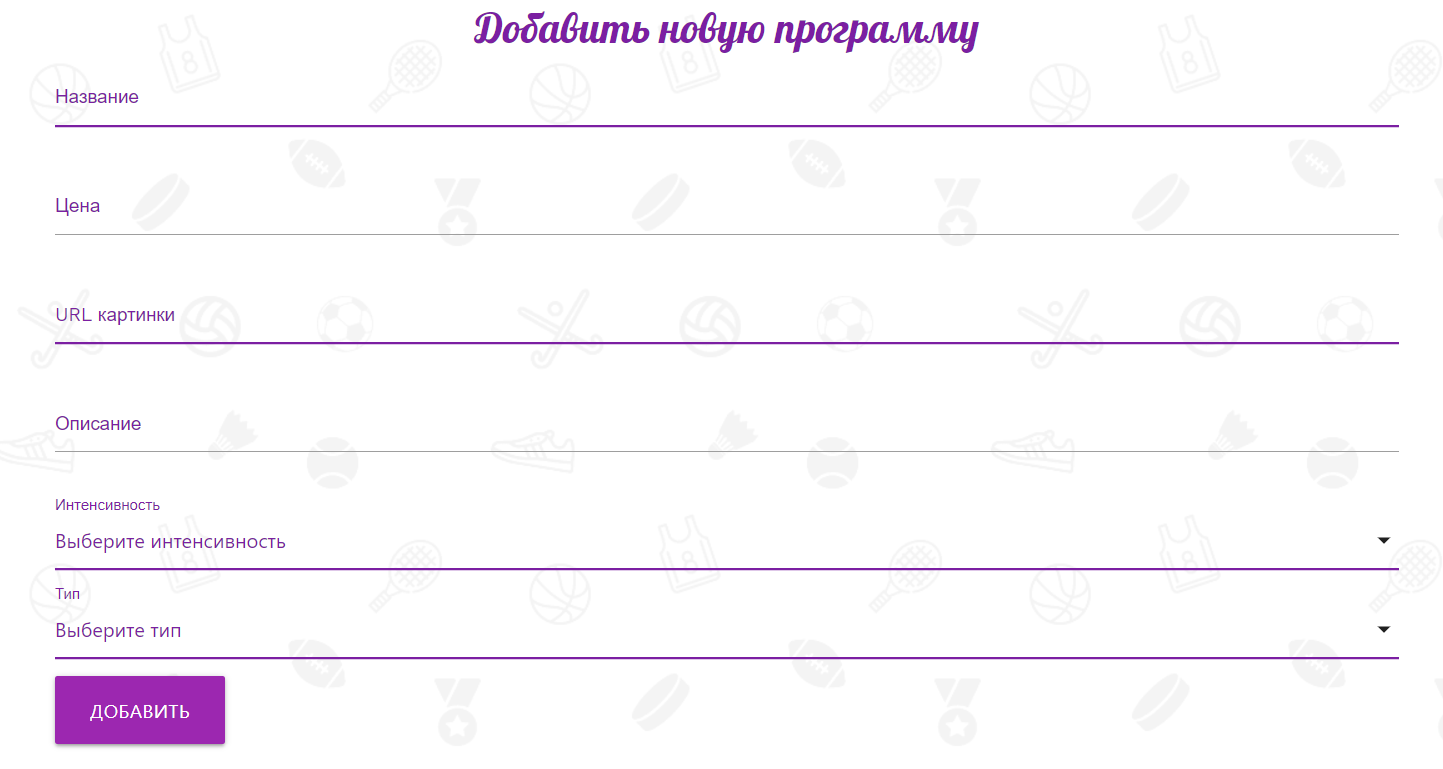
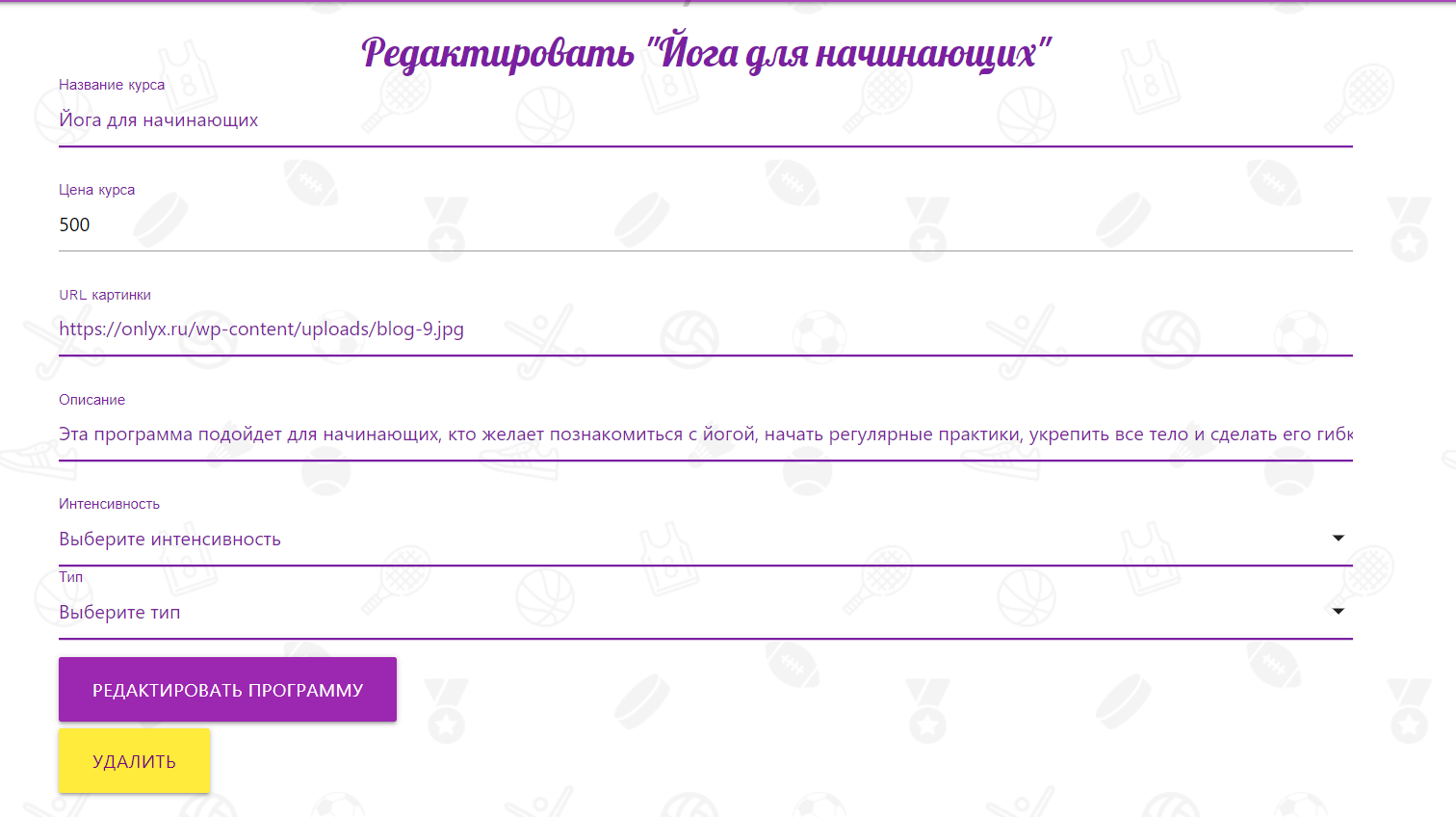
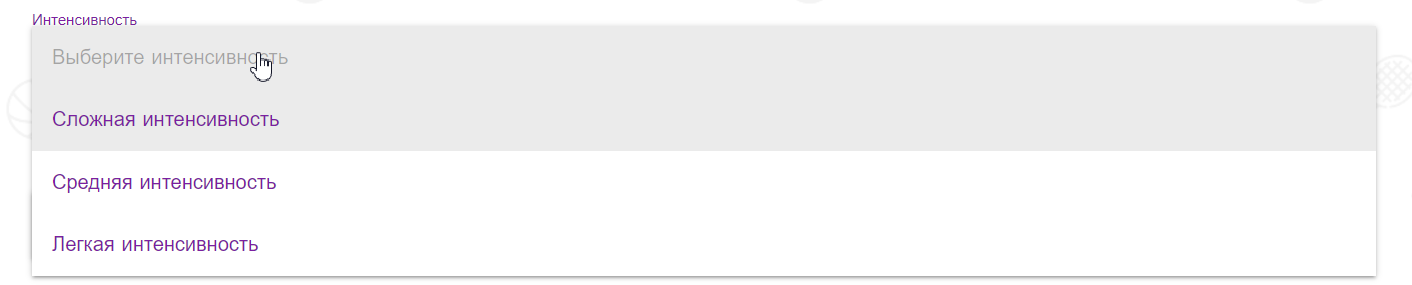
Формы, представленные на рисунках 3-4, позволяют администратору быстро создавать новые программы и обновлять информацию прямо на сайте, благодаря удобному и понятному интерфейсу, а также без использования других сервисов работы с базой данных и помощи разработчиков.

Рисунок 3 — Форма для добавления новой программы

Рисунок 4 — Форма для редактирования и удаления программы

При выборе интенсивности или типа открывается выпадающий список с возможностью выбора. В зависимости от выбора в базу данных отправляется определенное значение.

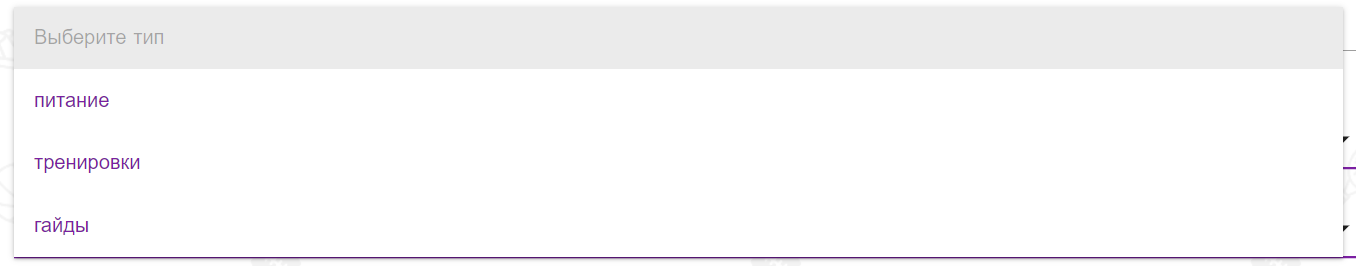
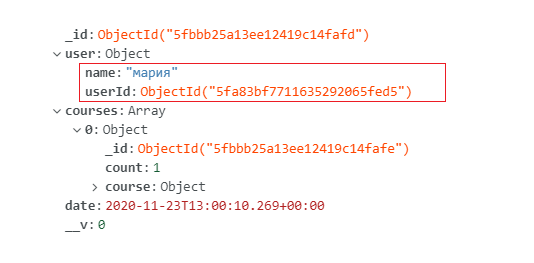
Рисунок 5 — Возможность выбора интенсивности

Рисунок 6 — Возможность выбора типа

Системой управления базой данных для хранения информации выбрана MongoDB. MongoDB — документоориентированная система управления базами данных. Для хранения данных используется JSON-подобный формат. Эта СУБД отличается высокой доступностью, масштабируемостью и безопасностью.

На схеме показано, как выглядит документ в MongoDB:

Рисунок 7—Пример документа «Orders»

MongoDB добавляет поле \_id с уникальным значением для идентификации документа в коллекции. Это поле обязательно для заполнения в каждом документе. Оно похоже на первичный ключ документа. Сами данные пользователя в заказе(name, userId) в MongoDB фактически хранятся как встроенный документ в самой коллекции, а в реляционных СУБД они обычно хранятся в отдельной таблице. Это одно из ключевых особенностей модели данных MongoDB.

Для интеграции MongoDB с проектом будем использовать Object Document Mapper (ODM) [Mongoose](https://mongoosejs.com/) для создания схем и моделей данных приложения. Это позволит нам организовать код приложения в соответствии с архитектурным шаблоном [модель-представление-контроллер (MVC)](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller), который позволяет отделить логику обработки вводимых пользователей данных и логику структурирования данных и их отображения для пользователя. Использование такого шаблона может упрощать будущие тестирование и разработку, реализуя разделение проблем в базе кода.

Mongoose выступает в качестве [внешнего](https://www.mongodb.com/what-is-mongodb) интерфейса для [MongoDB](https://www.mongodb.com/what-is-mongodb) , базы данных [NoSQL](https://en.wikipedia.org/wiki/NoSQL) с открытым исходным кодом, использующей документно-ориентированную модель данных. «Коллекция» «документов» в базе данных MongoDB [аналогична](https://docs.mongodb.com/manual/core/databases-and-collections/#collections) «таблице» «строк» ​​в реляционной базе данных.

Модели определяютсяс помощью Schema. Схема позволяет определять поля, хранящиеся в каждом документе, вместе с их требованиями к проверке и значениями по умолчанию. Также можно определить статические вспомогательные методы, для того, чтобы упростить работу с типами данных.

Затем схемы «компилируются» в модели с помощью этого mongoose.model() метода. Метод save() сохраняет текущий обьект в базу данных. Когда у нас есть модель, мы можем использовать ее для поиска, создания, обновления и удаления объектов данного типа.

Мы можем создавать ссылки из одного документа / экземпляра модели в другой, используя ObjectId поле схемы, или из одного документа во многие, используя массив ObjectIds. В поле хранится идентификатор связанной модели. Если нам нужно фактическое содержимое связанного документа, мы можем использовать [populate()](http://mongoosejs.com/docs/api.html#query_Query-populate) метод в запросе, чтобы заменить идентификатор фактическими данными.



Рисунок 8 — пример определения модели «Orders»

Мы можем искать записи, используя методы запроса, задав условия запроса в виде документа JSON.

Метод [find ()](http://mongoosejs.com/docs/api.html#query_Query-find)  получает все совпадающие записи. В качестве параметра в find передается объект, который устанавливает параметры фильтрации.

Следующие методы запрашивают одну запись:

* [findById()](http://mongoosejs.com/docs/api.html#model_Model.findById): Находит документ с указанным id(каждый документ имеет уникальный id).
* [findOne()](http://mongoosejs.com/docs/api.html#query_Query-findOne): Находит отдельный документ, соответствующий указанным критериям.
* [findByIdAndRemove()](http://mongoosejs.com/docs/api.html#model_Model.findByIdAndRemove), [findByIdAndUpdate()](http://mongoosejs.com/docs/api.html#model_Model.findByIdAndUpdate), [findOneAndRemove()](http://mongoosejs.com/docs/api.html#query_Query-findOneAndRemove), [findOneAndUpdate()](http://mongoosejs.com/docs/api.html#query_Query-findOneAndUpdate): Находит один документ, по id или критериям, обновляет или удаляет его.

Также будем использовать модуль [bcrypt](https://www.npmjs.com/package/bcrypt) для хеширования паролей **перед сохранением** в базу данных. Это необходимо в целях безопасности.

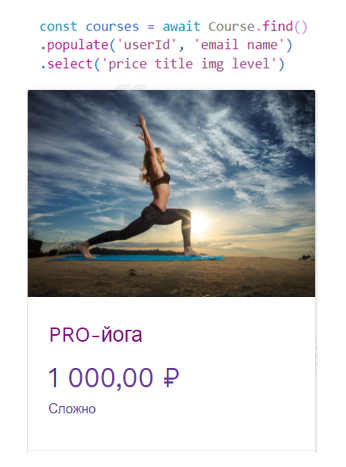


Рисунок 9 — Пример запроса получения массива программ

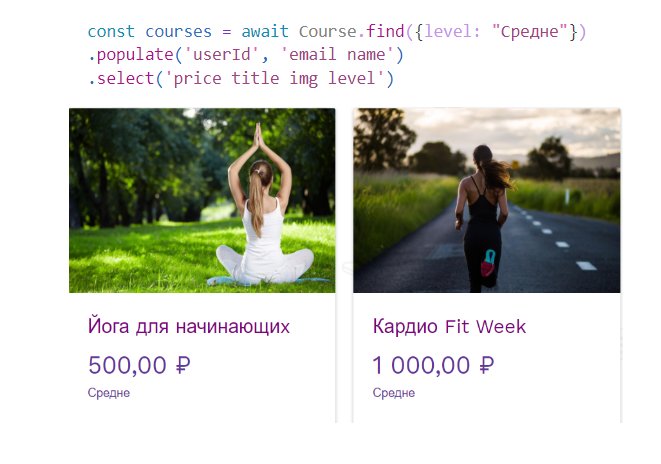


Рисунок 10 — Пример запроса получения массива программ с уровнем «Средне»

Рисунок 11 — Пример запроса получения курса с определенным id

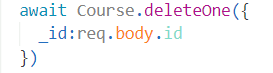


Рисунок 12 — Пример удаления одного документа по id из базы данных

# 3.2. Логическая схема базы данных

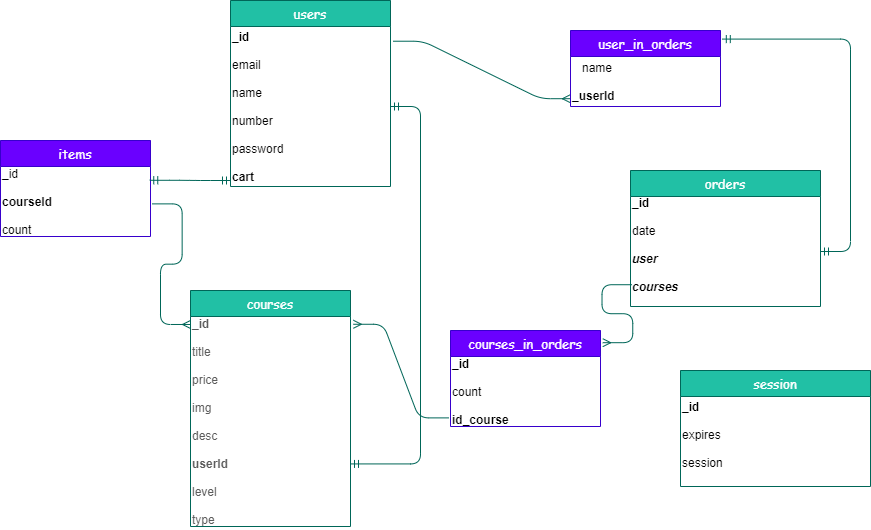


Рисунок 13 — Логическая схема

На рисунке 13 представлена логическая сxема базы данных. Так как MongoDB — документоориентированная система управления базами данных, то для хранения данных используется JSON-подобный формат.

Документы обозначаются на схеме зеленым. Фиолетовым цветом обозначаются встроенные документы, которые обеспечивают связи между несколькими документами.

Документ «users» содержит информацию о пользователе и встроенный документ «cart», который представлен массивом «items». Каждый элемент «items» содержит id программы и ее кол-во в корзине. Это обеспечивает связь с документом «courses» и позволяет получить полную информацию о выбранной программе (например, стоимость).

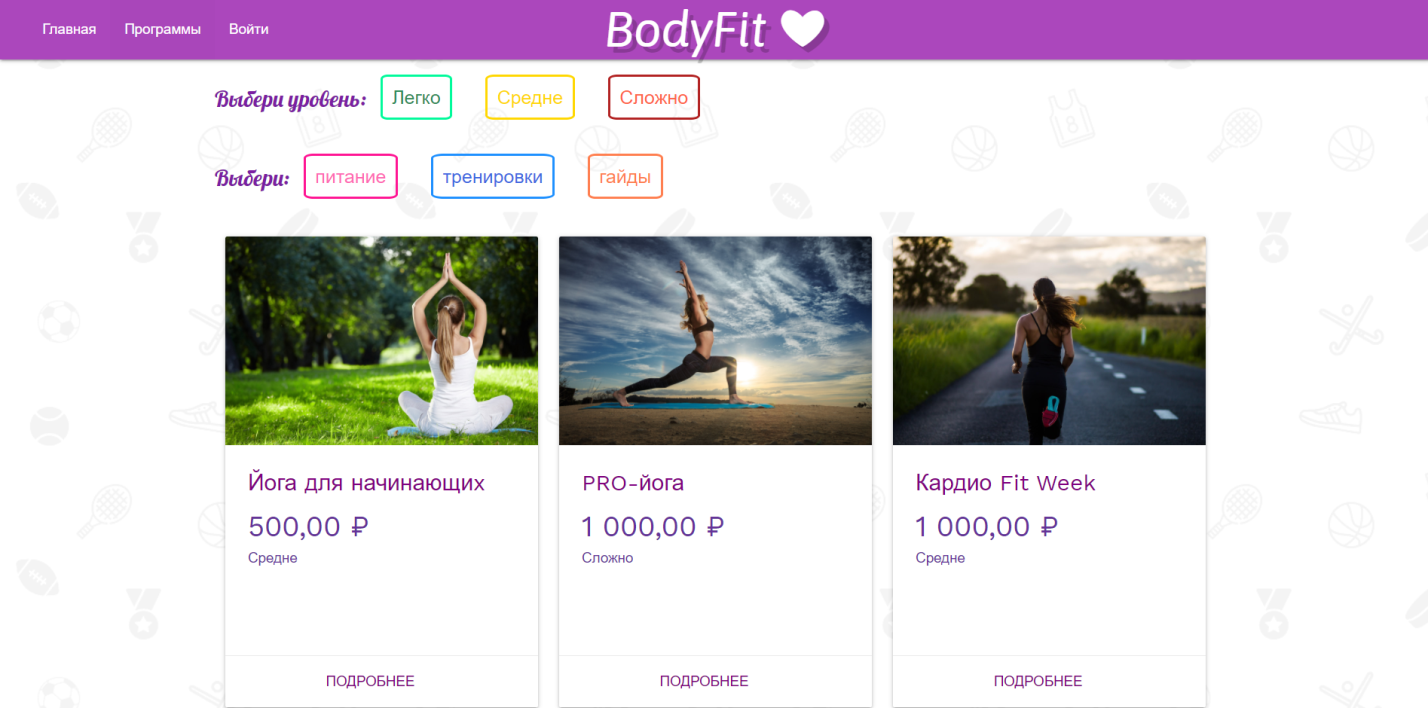
Документ «courses» хранит информацию о каждой программе. А также поле user\_id, которое всегда равно id администратора. Это позволяет администратору обновлять информацию о программах.

Документ «orders» содержит информацию о заказе, встроенный документ «user» и встроенный документ «courses». Встроенный документ «user» содержит имя и id пользователя, что позволяет получить полную информацию о пользователе в заказе.

Документ «session» хранит информацию о сессиях. Для поддержки сессий в MongoDB необходимо установить пакет [connect-mongodb](https://github.com/masylum/connect-mongodb)-session.

# Разработка интерфейса взаимодействия пользователя с системой

Для удобного взаимодействия пользователя был разработан следующий интерфейс.

Если пользователь является гостем, ему предоставляется возможность просмотра главной страницы, всех программ по интенсивности («Легко», «Средне» или «Сложно») или типу («Питание», «Тренировки» или «Гайды»), возможность регистрации или входа (для уже зарегистрированного пользователя).

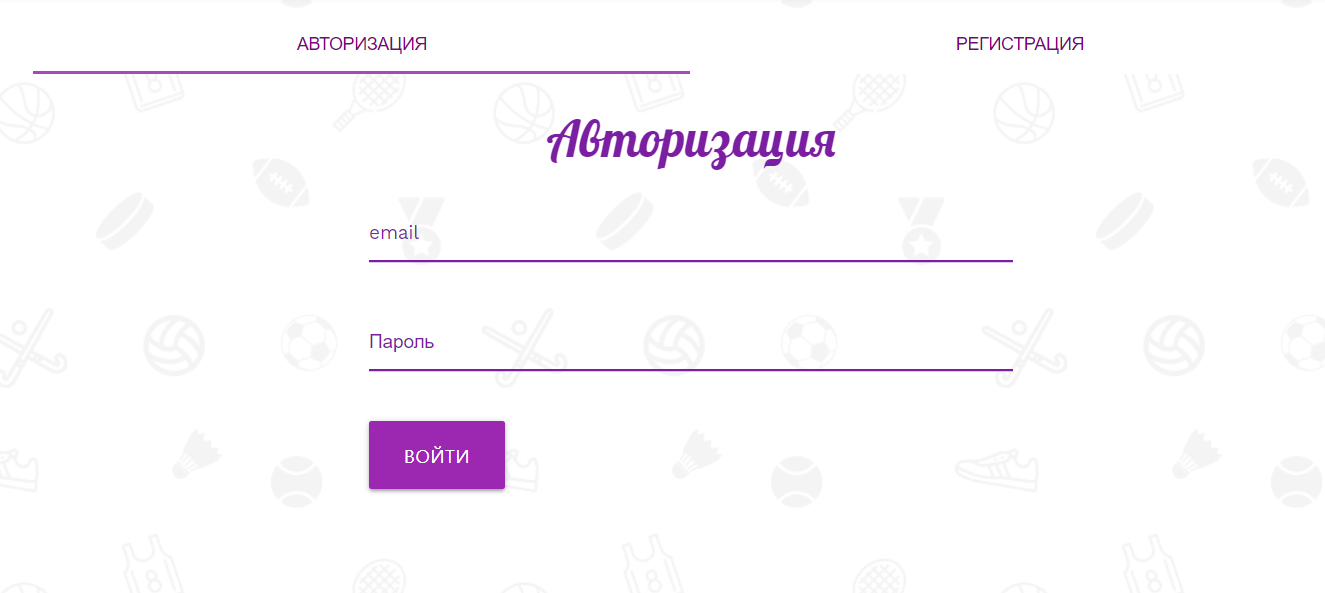
Рисунок 14 — Страница всех программ (для гостя)

Рисунок 15 — Страница «Авторизация»

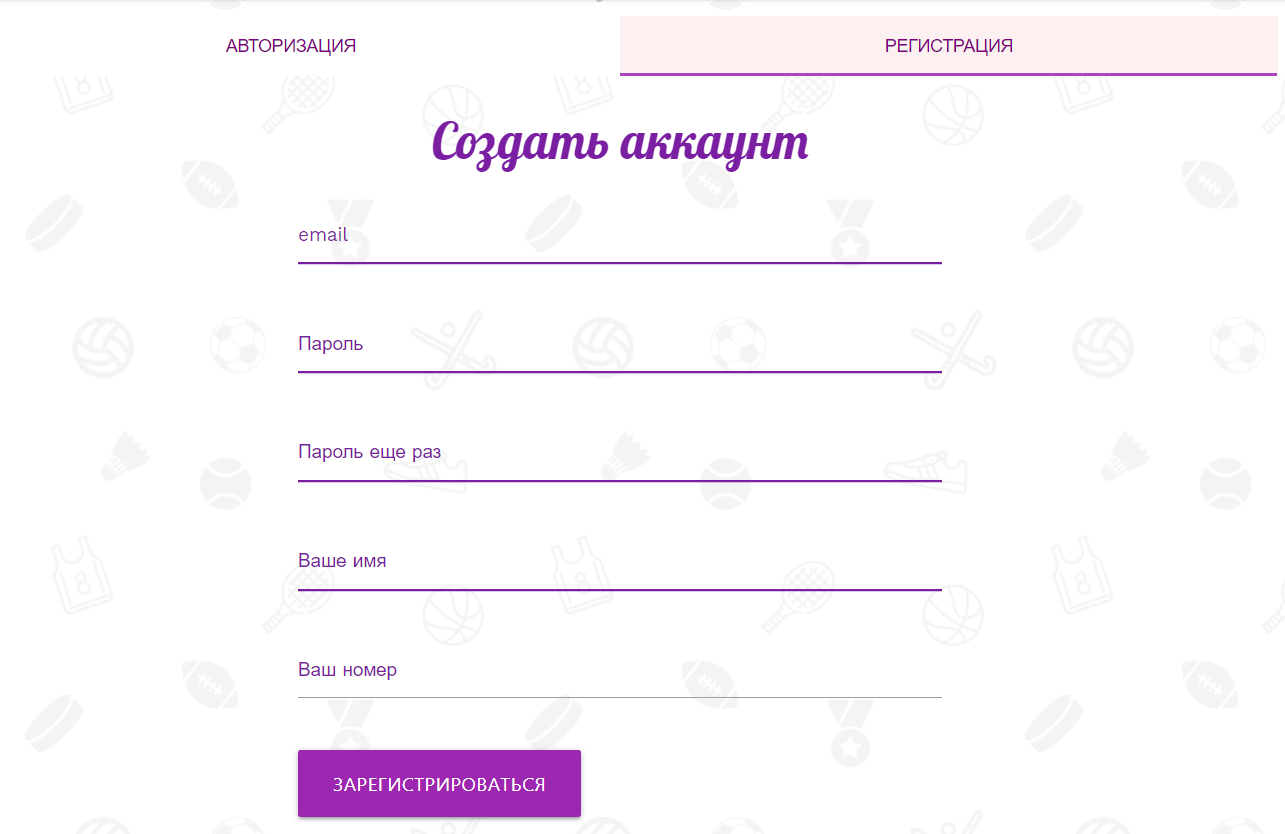


Рисунок 16 — Страница «Регистрация»

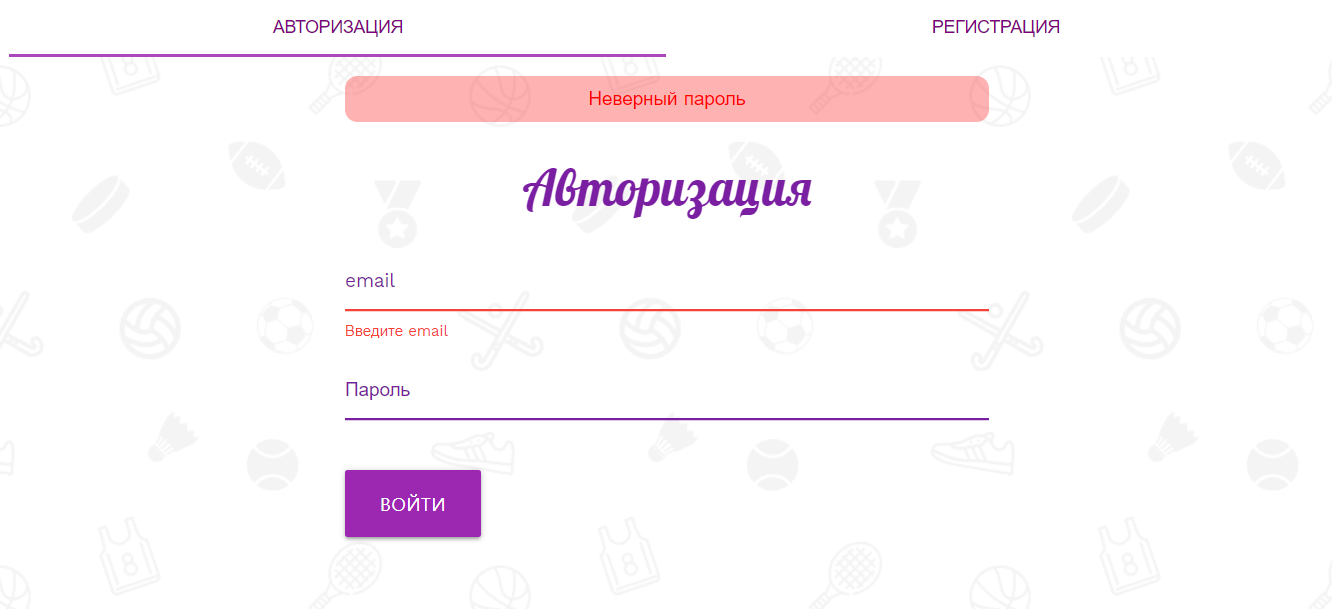
Если данные были введены некорректно, пользователю выводится сообщение об ошибке. Например, указан неправильный email (неправильное написание, отсутствие данного домена или такой email уже существует), номер, или пароли не совпадают при регистрации. При входе указан незарегистрированный email или неправильный пароль. Сообщение об ошибке содержит информацию о конкретном неправильном поле.

Рисунок 17 — Возникновение ошибки на странице «Авторизация»

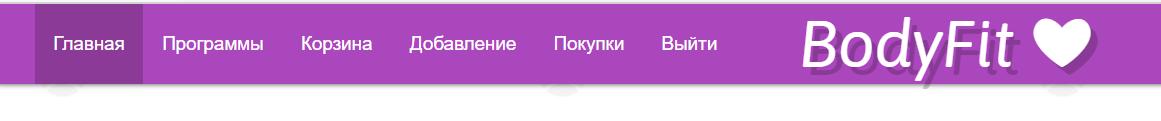
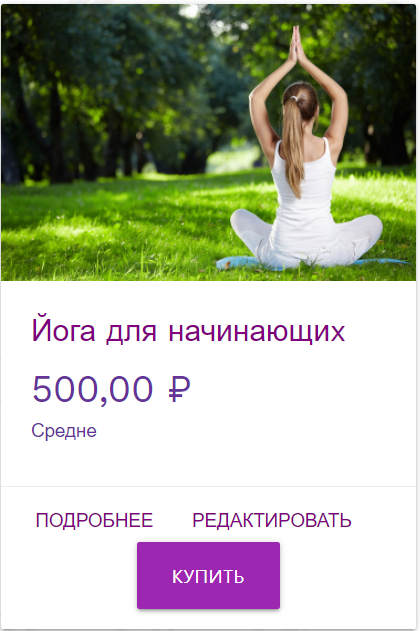
Если пользователем является администратор, ему доступна страница добавления, редактирования и удаления.

Рисунок 18 — Появление вкладки «Добавления»



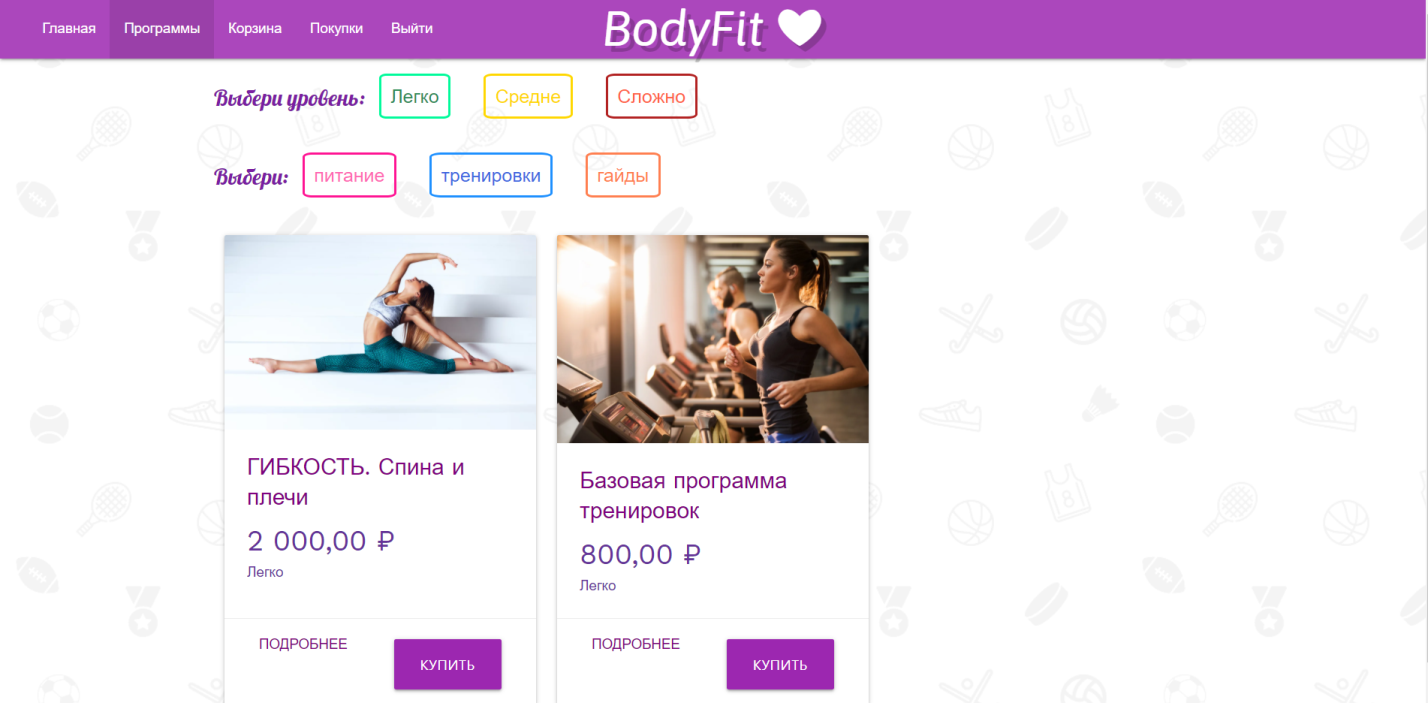
Рисунок 19 — Появление возможности редактирования

Рисунок 20 — Страница программ с уровнем «Легко»

Если пользователь зарегистрировался или зашел, ему доступна функция совершения покупки, а также просмотра именно своих заказов, которые он совершил на протяжении всего времени использования данного приложения.

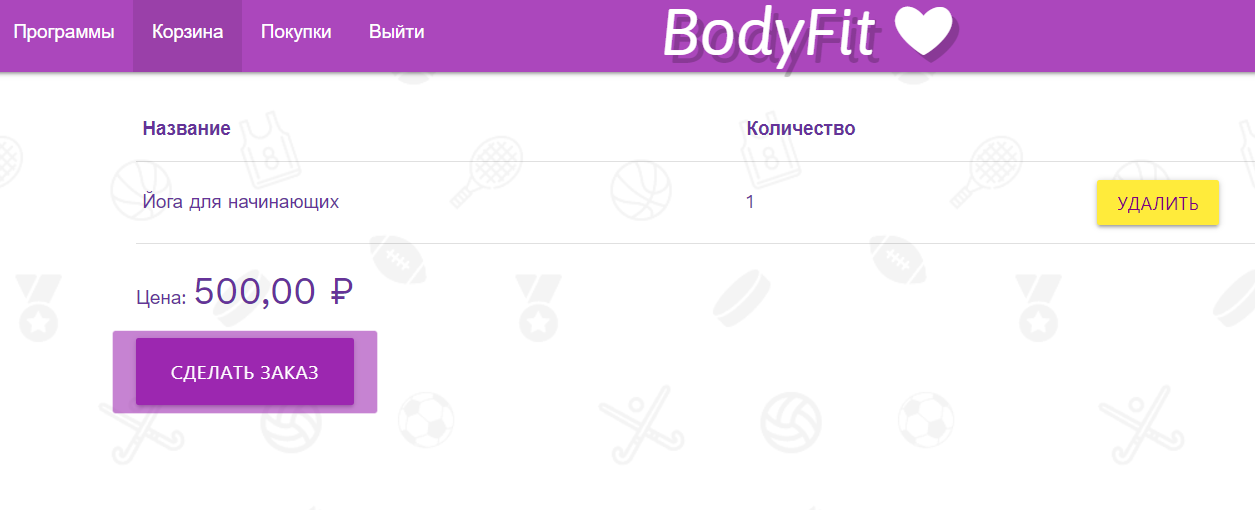
При нажатии пользователем на кнопку «Купить» в корзину добавляется программа. У пользователя появляется возможность просмотра всех программ в корзине, удаления программы из корзины или совершения покупки.

Рисунок 21 — Страница «Корзины»

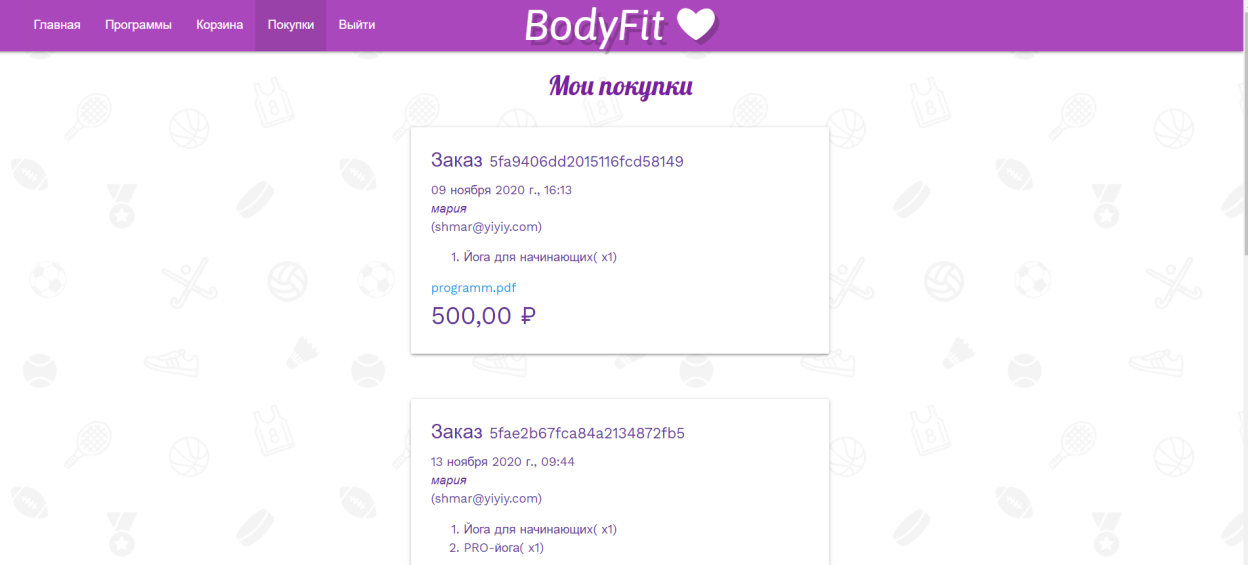
При нажатии на кнопку «Сделать заказ» во вкладке «Покупки» пользователь имеет возможность просмотра всех своих покупок.

Рисунок 22 — Страница «Покупки»

# 3.4. Разработка архитектуры приложения

Для разработки данного приложения используется паттерн MVC.

Паттерн MVC включает ряд компонентов:

* **Модели** определяют структуру и логику используемых данных.
* **Представления** (views) определяет визуальную часть, как данные будут отображаться.
* **Контроллеры** обрабатывают входящие http-запросы, используя для обработки модели и представления, и отправляет в ответ клиенту некоторый результат обработки, нередко в виде html-кода.
* **Система маршрутизации** как дополнительный компонент сопоставляет запросы с маршрутами и выбирает для обработки запросов определенный контроллер.

В общем случае, когда к приложению приходит запрос, система маршрутизации выбирает нужный контроллер для обработки запроса. Контроллер обрабатывает запрос. В процессе обработки он может обращаться к данным через модели и для рендеринга ответа использовать представления. Результат обработки контроллера отправляется в ответ клиенту. Ответ представляет html-страницу, которую пользователь видит в своем браузере.

Организация взаимодействия приложения с сервером посредством протокола HTTP осуществляется за счет принципа REST (Representational state transfer).

REST работает на уровне HTTP и использует нижеприведённые методы для работы с ресурсами на сервере:

POST — для создания ресурса;

GET — для его получения;

PUT — для его обновления;

DELETE — для его удаления.

В данном случае, например, необходимо иметь возможность добавлять новые программы, просматривать все программы и просматривать все заказы.

Эти операции соответствуют следующим маршрутам:

* POST /add— создание новой программы;
* GET /courses/:id — получение одной программы;
* GET /courses — получение всех программ.

# 4. Проектно-технологическая часть

# 4.1. Проектирование начального и тестового наполнения базы данных. Порядок развертывания системы.

Процесс наполнения базы данных описан в пункте 3.1. данной работы с примерами форм для удобства пользования.

Самая простая процедура развертывания сайта состоит из следующих этапов:

1)Перемещение файлов на сервер;

2)Запуск команды npm install для установки новых зависимостей;

3)Запуск node. В данном случае командой npm run start.

Также можно развернуть приложение на сервер PaaS-провайдера (Heroku), либо с помощью Docker.

Данная система тестировалась в следующих браузерах: Google Chrome, Firefox, Microsoft Edge.

# Заключение

При выполнении курсовой работы на тему «Интернет магазин фитнес-программ» была исследована и описана предметная область, проведен анализ аналогов данной системы, а также выбор инструментов и платформы для разработки.

Проведен анализ объектов автоматизации и разработаны методы решения технических задач. Также была разработана и реализована структура базы данных, серверная часть приложения и веб-интерфейс системы. База данных была наполнена тестовыми данными.

Результатом данной курсовой работы является рабочий интернет-магазин фитнес-программ.

# Список использованных источников

1. Арно Лоре Проектирование веб-API / Пер. с англ. Д. А. Беликова.– М.: ДМК Пресс, 2020.– 440 с

2. Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. - СПб.: Питер, 2013. - 240 c.

3. Янг А., Мек Б., Кантелон М. Я Node.js в действии. 2-е изд. — СПб.: Питер, 2018. — 432 с.: ил. — (Серия «Для профессионалов»).

4. Ревунков Г.И., Ковалева Н.А., Силантьева Е.Ю. Проектирование баз данных. [Электронный ресурс] – МГТУ им. Н.Э.Баумана, 2018. – 48 с. Режим доступа: <https://e.lanbook.com/reader/book/103499/>

5. https://nodeguide.ru

6. Руководство по программированию на JavaScript [Электронный ресурс] – режим доступа: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express\_Nodejs/skeleton\_website

7. Изучаем Node.js [Электронный ресурс] – режим доступа: <https://metanit.com/web/nodejs/7.1.php>

8. Learning JavaScript Design Patterns — Addy Osmani 2020 – 439c.