Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Системный анализ и проектирование информационных систем с модулем веб-приложения»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Старший преподаватель кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Ю. Петрович |
|  | \_\_\_.\_\_\_\_.2024 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«СЕРВИС ОЦЕНКИ КИНОФИЛЬМОВ»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 272303  ВДОВЕНКО Никита Дмитриевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2024  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2024

ВВЕДЕНИЕ

С развитием интернета и цифровых технологий наш мир становится все более цифровым и информатизированным. В таких условиях нет сферы, которая бы осталась в стороне от влияния технологического прогресса. В контексте развлекательной индустрии кинематограф, безусловно, занимает важное место. Однако среди огромного количества кинопродукции, доступной для зрителей, не всегда легко выбрать фильм, который действительно соответствует их вкусам и предпочтениям. В таких условиях сервисы оценки кинофильмов становятся неотъемлемой частью киноландшафта, предоставляя зрителям возможность получить объективные оценки и рекомендации от других пользователей перед просмотром фильма.

Сервис оценки кинофильмов является цифровой платформой, предназначенной для сбора, анализа и отображения оценок и отзывов о фильмах со стороны пользователей. Этот сервис обеспечивает зрителей информацией о качестве и интересности фильмов, помогая им принимать более обоснованные решения относительно выбора кинопродукции для просмотра.

В данной работе будет рассмотрен процесс разработки и функциональные возможности сервиса оценки кинофильмов. В частности, будет проанализирована предметная область, определены основные требования к сервису, разработаны методы сбора и анализа данных о кинофильмах, а также рассмотрены возможные алгоритмы для формирования рейтингов и рекомендаций.

Целью данной работы является создание эффективного и удобного в использовании сервиса оценки кинофильмов, который предоставит пользователям актуальную и достоверную информацию о фильмах, помогая им принимать обоснованные решения при выборе кинопродукции для просмотра.

Задачи работы:

* ознакомиться с предметной областью;
* проанализировать процессы;
* разработать пояснительную записку и код.

Объектом исследования данной курсовой работы является создание программного средства для оценки и анализа кинофильмов.

# **АНАЛИЗ И МОДЕЛИРОВАНИЕ ДЕЯТЕЛЬНОСТИ СЕРВИСА ОЦЕНКИ КИНОФИЛЬМОВ**

## **Описание деятельности** **сервиса оценки кинофильмов**

Сервис оценки кинофильмов представляет собой платформу, которая обеспечивает пользователям доступ к информации о фильмах, их оценкам и отзывам со стороны других пользователей. В основе деятельности такого сервиса лежит сбор, анализ и предоставление обработанных данных о кинопродукции для помощи пользователям в выборе фильмов для просмотра.

Одним из ключевых аспектов деятельности сервиса оценки кинофильмов является сбор информации о фильмах. Это включает в себя получение данных о новых кинопремьерах, а также обновление информации об уже существующих фильмах. Для этого могут использоваться различные источники данных, такие как кинотеатральные прокаты, онлайн-платформы для просмотра фильмов, кинематографические базы данных и рецензии от кинокритиков.

Дальнейшим шагом в работе сервиса является анализ полученных данных о кинофильмах. Это включает в себя оценку критиков и пользователей, анализ трендов и популярности фильмов, а также формирование рейтингов и рекомендаций на основе этих данных. Для этого могут применяться различные алгоритмы и методы машинного обучения, направленные на выявление предпочтений пользователей и предоставление персонализированных рекомендаций.

Организация работы сервиса оценки кинофильмов также включает в себя поддержание актуальности и обновление информации о фильмах. Это может включать в себя добавление новых кинопремьер, коррекцию рейтингов и отзывов на основе новой информации, а также удаление устаревших данных.

Управление интерфейсом и пользовательским опытом также важные аспекты деятельности сервиса оценки кинофильмов. Это включает в себя разработку удобного и интуитивно понятного пользовательского интерфейса, обеспечение возможности поиска и фильтрации фильмов по различным критериям. а также предоставление дополнительных функций, таких как сохранение списков любимых фильмов или создание персональных рекомендаций на основе предпочтений пользователя.

Для обеспечения качественного обслуживания пользователей сервис оценки кинофильмов также может заботиться о мониторинге обратной связи от пользователей, оперативном реагировании на жалобы и претензии, а также проведении опросов и исследований для выявления потребностей и предпочтений аудитории.

Таким образом, сервис оценки кинофильмов представляет собой комплексную платформу, цель которой состоит в обеспечении пользователей информацией о качестве и интересности кинопродукции, помогая им принимать обоснованные решения при выборе фильмов для просмотра.

## **Разработка функциональной модели деятельности сервиса оценки кинофильмов в нотации IDEF0**

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.

Рассмотрим информационную систему учета, выполненную с помощью средств моделирования функций IDEF0. Для начала необходимо сделать контекстную модель информационной системы. Контекстная диаграмма – самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой.

На рисунке 1.2.1 представлена диаграмма IDEF0, где изображен основной блок «Оценить кинофильм», что является главной целью процесса.

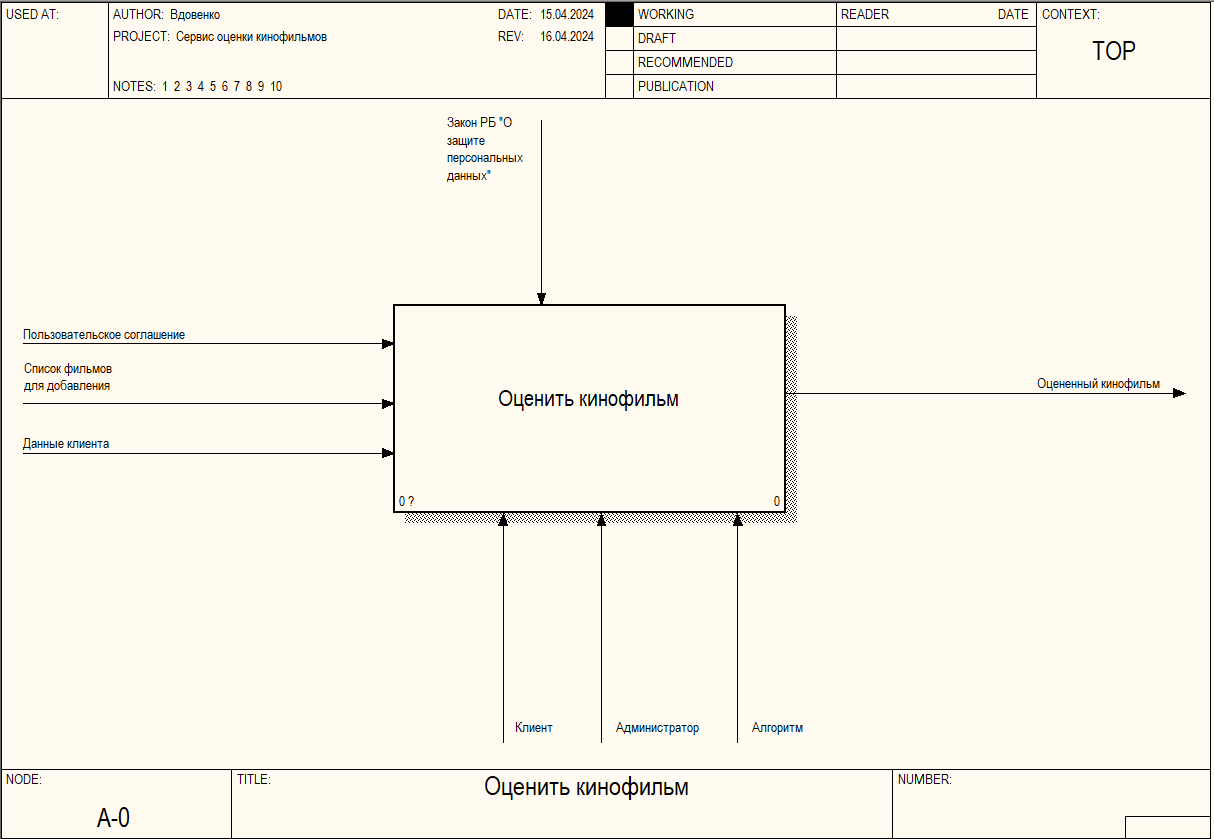


Рисунок 1.2.1 – Контекстная диаграмма «Оценить кинофильм» модели A-0

На рисунке 1.2.2 представлена декомпозиция процесса автоматизации работы таксопарка. В данном случае процесс происходит в три этапа: «Добавить кинофильм в каталог», «Посмотреть кинофильм», «Поставить оценку кинофильму».

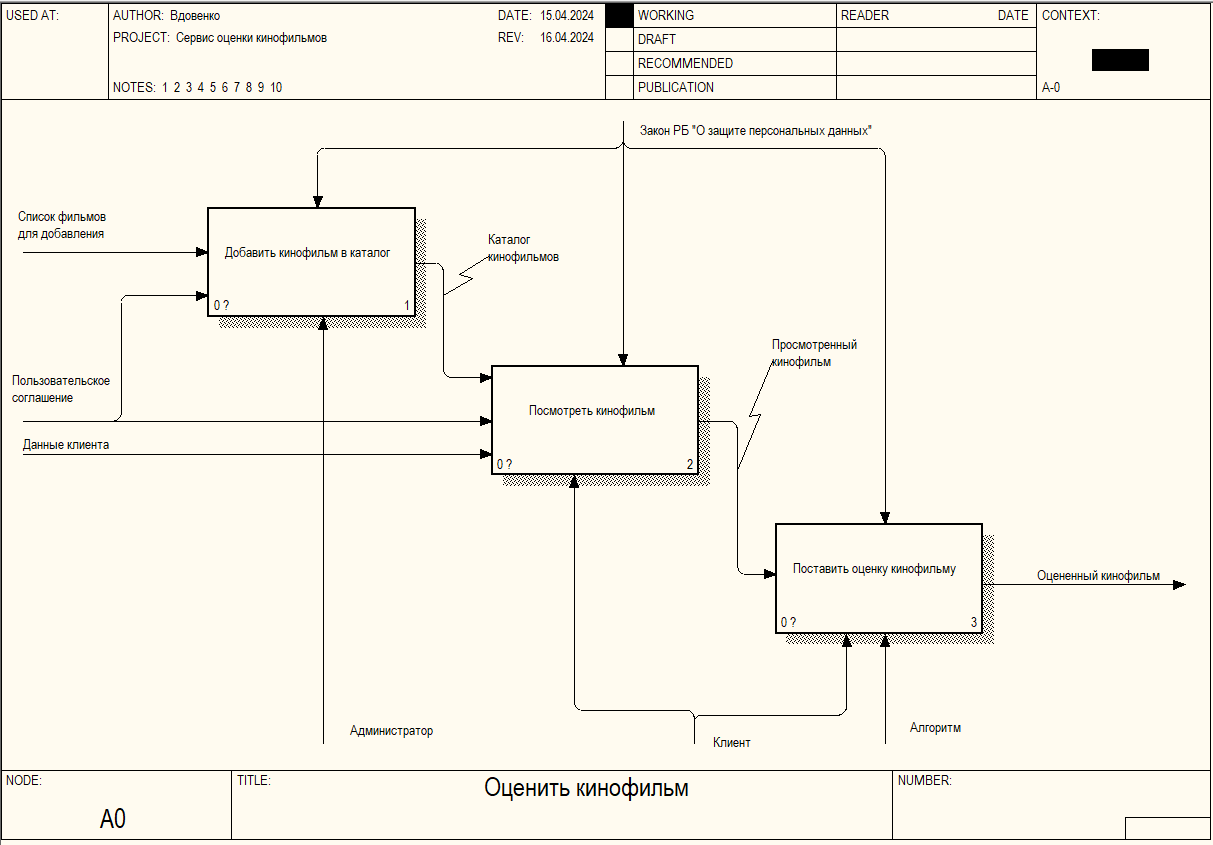


Рисунок 1.2.2 – Декомпозиция модели «Оценить кинофильм» уровня A0

На рисунке 1.2.3 представлена декомпозиция «Поставить оценку кинофильму». Основные этапы: «Проанализировать просмотр кинофильма», «Выбрать оценку кинофильму», «Агрегировать оценки кинофильма».

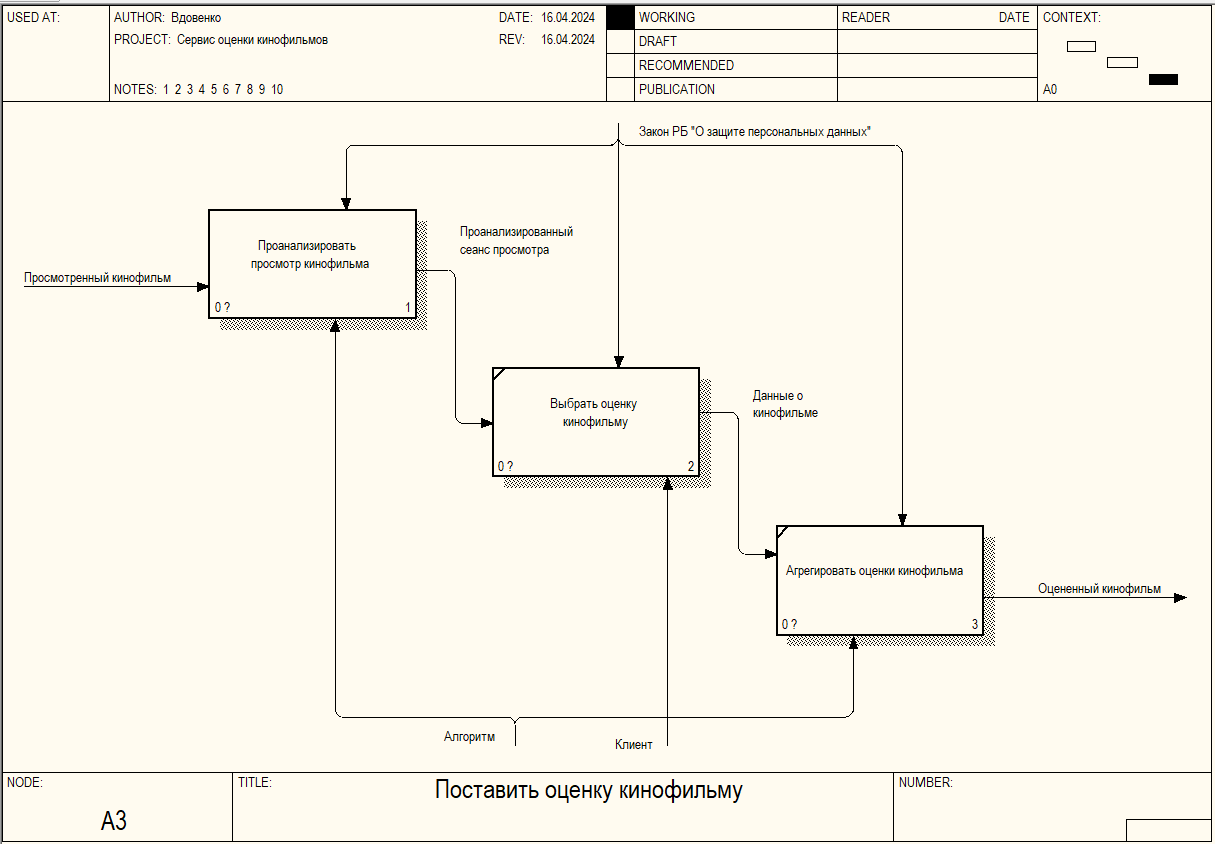


Рисунок 1.2.3 – Декомпозиция модели «Поставить оценку кинофильму» уровня A3

На рисунке 1.2.4 представлена декомпозиция модели «Проанализировать просмотр кинофильма», где главными этапами являются: «Проанализировать время чтения описания кинофильма», «Проанализировать время просмотра кинофильма», «Агрегировать метрики анализа, которые переходят от одного к другому поэтапно.

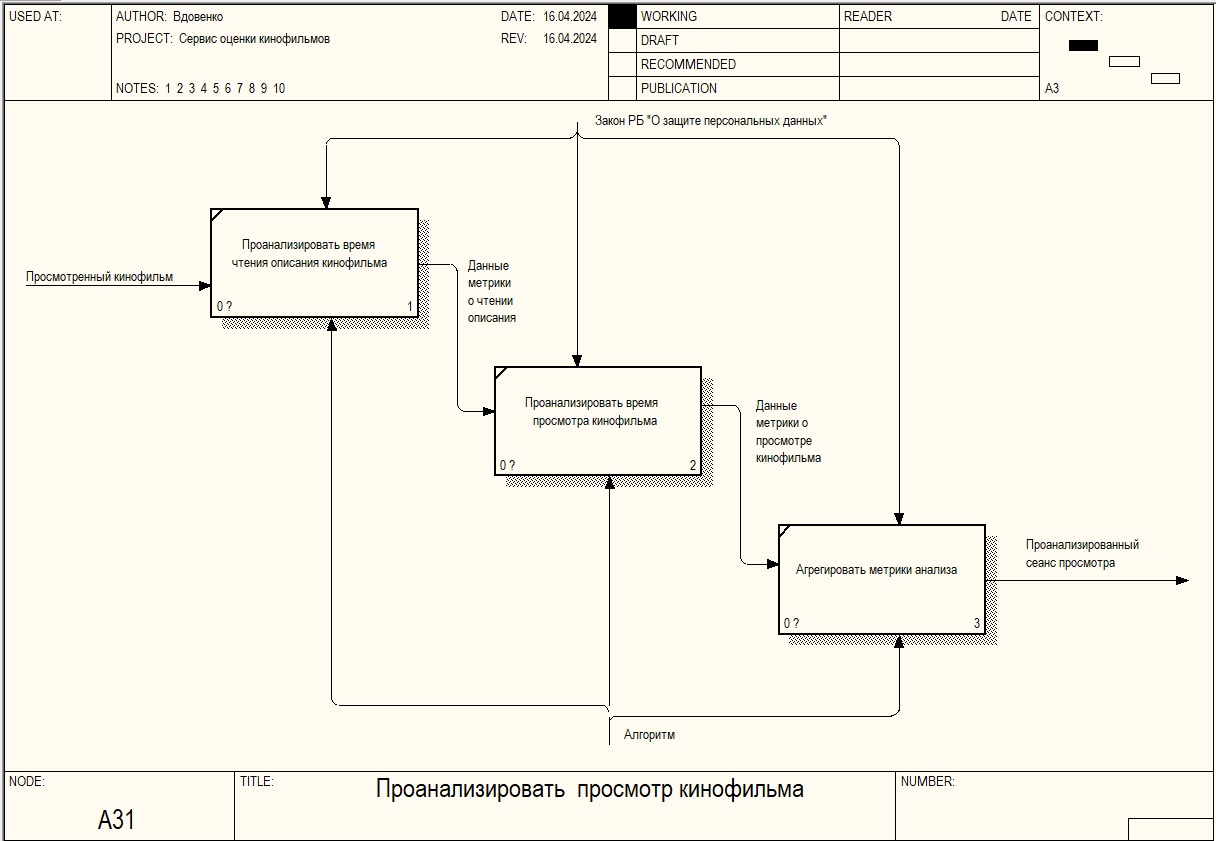


Рисунок 1.2.4 – Декомпозиция модели «Проанализировать просмотр кинофильма» уровня A31

Таким образом, выше представлены три уровня декомпозиции главного процесса, который звучит как «Оценить кинофильм», что создается для того, чтобы разбить главный процесс на подпроцессы скольких угодно уровней. Это помогает обозначить для разработчика более конкретизированную последовательность действий и подробно рассмотреть каждый этап разработки, включая входные и выходные данные, а также механизмы и управление. Нотация IDEF0 дает возможность более точно и подробно изучить данные процессы и составляющие их аспекты.

## **Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований.**

При разработке и проектировании программного средства будут использованы следующие технологии:

1. Для реализации серверной (back-end) части программного средства – язык программирования Node.js.
2. Для реализации клиентской части (front-end) части программного средства – язык программирования JavaScript (практически единственное решения для реализации клиентской части веб-приложений).
3. Для организации хранения данных используется облачная СУБД MongoDB Atlas.
4. Для разметки используется язык гипертекстовой разметки HTML и шаблонизатор EJS.
5. Для стилистического оформления используются каскадные таблицы стилей CSS.

Для перехода к разработке программного средства необходимо определить функциональные требования.

Диаграмма вариантов использования для данного курсового проекта представлена на рисунке 1.3.1. Там отображены возможности использования приложения.

Актёр «клиент» представляет собой человека, прошедшего авторизацию в системе.

Клиент имеет возможность осуществлять:

* Оценить кинофильм;
* Написать рецензию к кинофильму;
* Оценить рецензии других пользователей.

Актёр «администратор» представляет собой человека, также прошедшего авторизацию в системе.

* Добавить кинофильм в каталог;
* Изменить информацию о кинофильме;
* Удалить кинофильм;
* Удалить рецензию клиента;

Оба актёра «клиент» и «администратор» могут осуществлять:

* Посмотреть информацию о кинофильме;
* Прочитать рецензии других пользователей;
* Зарегистрироваться/авторизоваться;
* Работать с данными в профиле;
* Найти кинофильм по названию;
* Отсортировать кинофильмы по параметрам.

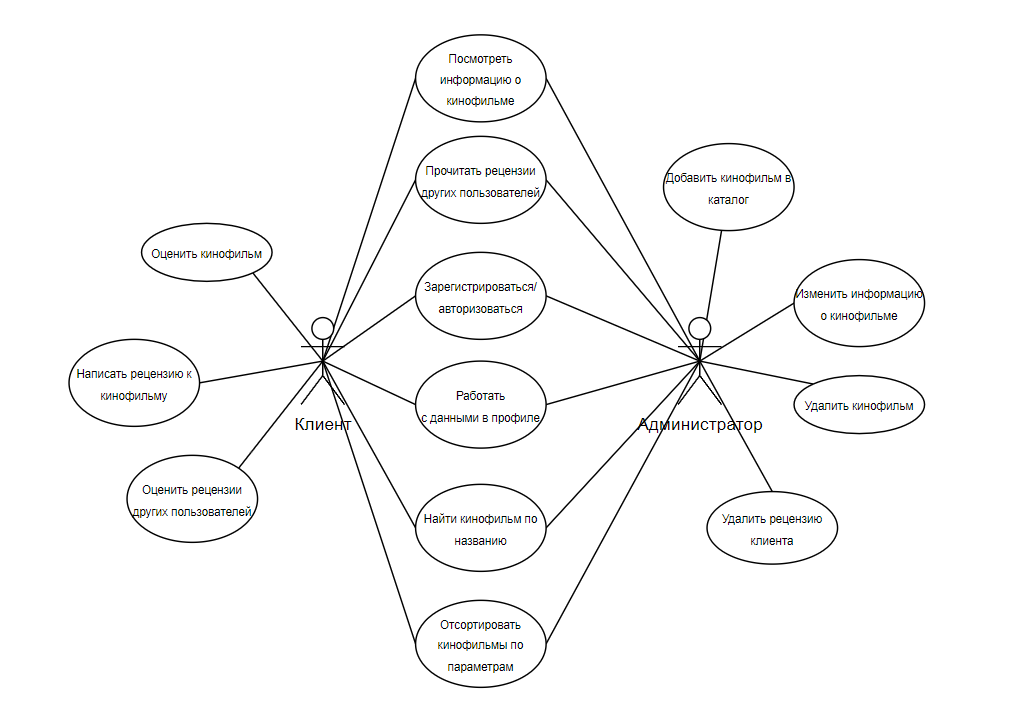


Рисунок 1.3.1 – Диаграмма вариантов использования

Проектирование диаграммы вариантов использования является важным этапом, позволяющим структурировать функциональность приложения. Эта диаграмма предоставляет детальный обзор взаимодействия пользователей с системой в различных сценариях использования. Каждый вариант использования тщательно разрабатывается с учетом особенностей и требований конкретной роли, что обеспечивает полное и ясное представление о том, как пользователи будут взаимодействовать с приложением. Такой подход способствует более глубокому пониманию функционала приложения с точки зрения конечных пользователей и позволяет лучше адаптировать продукт к их уникальным потребностям и задачам.

**1.4 Разработка информационной модели предметной области**

Каждая информационная система в зависимости от ее назначения имеет дело с той или иной частью конкретного мира, которую принято называть предметной областью информационной системы.

Анализ предметной области является необходимым начальным этапом разработки любой информационной системы. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных.

Информационный объект – это описание некоторой сущности предметной области реального объекта, процесса, явления или события.

Информационный объект (сущность) образуется совокупностью логически взаимосвязанных атрибутов (свойств), представляющих качественные и количественные характеристики объекта (сущности).

Информационная модель данных предназначена для представления семантики предметной области в терминах субъективных средств описания - сущностей, атрибутов, идентификаторов сущностей, супертипов, подтипов и т.д.

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы.

Перечень и описание сущностей информационной модели IDEF1X (рисунок 1.4.1):

* Пользователь;
* Клиент;
* Водитель;
* Машина;
* Поездка;
* Заказ.

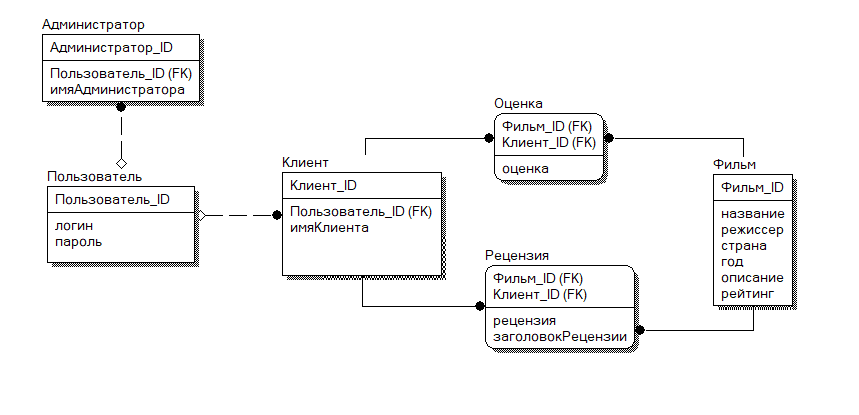


Рисунок 1.4.1 – Информационная модель системы в нотации IDEF1.X

Сущность Пользователь содержит информацию о зарегистрированном пользователе. Она включает в себя следующие атрибуты:

* «Пользователь\_ID» – первичный ключ таблицы, однозначно идентифицирующий запись;
* «логин» – логин пользователя;
* «пароль» – пароль пользователя.

Сущность Клиент содержит информацию о клиенте. Она включает в себя следующие атрибуты:

* «Клиент\_ID» – первичный ключ таблицы;
* «имяКлиента».

Сущность Администратор содержит информацию об администраторе. Она включает в себя следующие атрибуты:

* «Администратор\_ID» – первичный ключ таблицы;
* «имяАдминистратора»;
* «номерТелефонаВодителя».

Сущность «Фильм» содержит информацию о фильмах, доступных для оценки. Атрибуты сущности Она включает в себя следующие атрибуты:

* «Фильм\_ID» – первичный ключ таблицы;
* «название»;
* «режиссер»;
* «страна»;
* «год»;
* «описание»;
* «рейтинг».

Сущность «Оценка» предназначена для хранения оценок фильмов пользователями. Она включает в себя следующие атрибуты:

* «Фильм\_ID» – внешний ключ, указывающий на фильм, который оценивается;
* «Клиент\_ID»­ – внешний ключ, ссылающийся на пользователя, который оставил оценку;
* «оценка».

Сущность Рецензия содержит информацию о заказе. Она включает в себя следующие атрибуты:

* «Фильм\_ID» – внешний ключ, указывающий на фильм, который оценивается;
* «Клиент\_ID»­ – внешний ключ, ссылающийся на пользователя, который оставил оценку;
* «заголовокРецензии»
* «рецензия».

Таким образом, информационные системы играют важную роль в современном мире, обеспечивая эффективное управление информацией в различных сферах деятельности. Они помогают анализировать предметные области, определять информационные потребности пользователей и формировать содержание баз данных. Благодаря информационным моделям данных, таким как IDEF1X, можно наглядно представить семантику предметной области, что облегчает понимание структуры данных и их взаимосвязей. Такие системы обеспечивают эффективное управление данными, что является ключевым элементом в достижении успеха в современном информационном обществе.

**1.5 UML-модели представления программного средства и их описание**

UML (с английского аббревиатура расшифровывается как Unified Modeling Language – унифицированный язык моделирования) – это способ наглядно описать архитектуру, проектирование и реализацию комплексных программных систем. Код типичного приложения включает в себя тысячи строк, за связями и иерархиями которых очень непросто уследить. С помощью диаграмм UML структуру программы можно разделить на компоненты и подкомпоненты.

Диаграмма последовательности (англ. sequence diagram) – UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках прецедента.

На рисунке 1.5.1 представлена диаграмма последовательности.

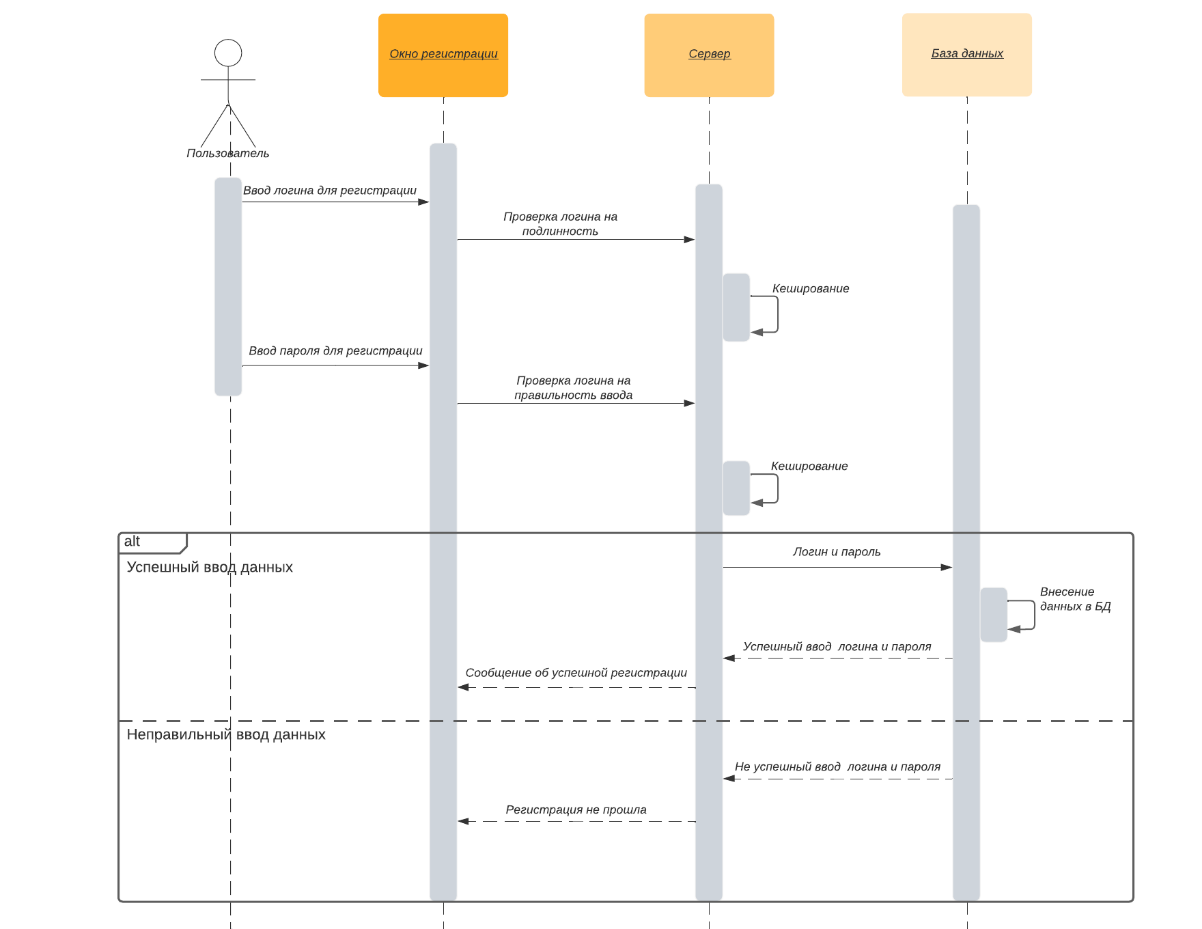


Рисунок 1.5.1 – UML-диаграмма последовательности, процесс «Регистрация»

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

На рисунке 1.5.2 представлена диаграмма развертывания.

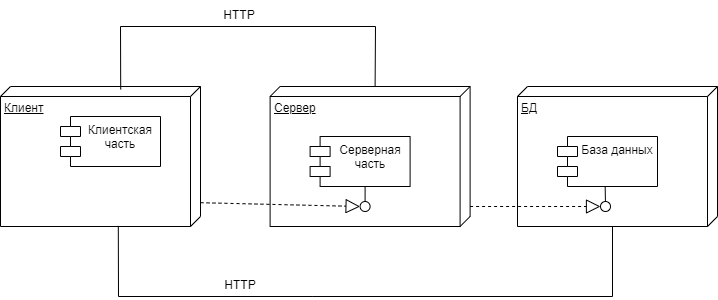




Рисунок 1.5.2 – UML-диаграмма развертывания

Диаграмма компонентов – это типа UML-диаграммы, которая используется для визуализации организации компонентов системы и зависимостей между ними, что позволяют получить высокоуровневое представление о компонентах системы.

На рисунке 1.5.3 представлена диаграмма компонентов.

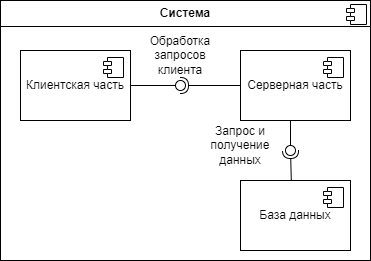


Рисунок 1.5.3 – UML-диаграмма компонентов

UML-модели представления программного средства играют важную роль в разработке и документировании программных систем. С помощью таких моделей разработчики могут визуализировать архитектуру системы, документировать требования к ней и обеспечить лучшее понимание структуры и взаимодействия компонентов. Диаграммы последовательности, развертывания и компонентов являются основными инструментами UML, которые позволяют эффективно моделировать различные аспекты программного обеспечения. Эти модели не только упрощают процесс разработки, но и способствуют более качественному пониманию системы всеми участниками проекта.

# **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**



## **Постановка задачи**

Основная задача курсового проекта – разработать веб-приложение для сервиса оценки кинофильмов.

Программное средство должно обладать следующими основными функциями, предоставляемые клиенту:

* регистрация и авторизация;
* просмотр информации о фильмах;
* оценка кинофильмов;
* добавление кинофильмов в избранное;
* написание рецензий о фильмах.

Для администратора должны быть добавлены следующие функции:

* авторизация;
* управление кинофильмами (добавление, удаление, редактирование);
* просмотр информации о фильмах;
* модерация рецензий.

Также программное средство должно быть реализовано в виде веб-сайта с использованием современных веб-технологий. Необходимо создать базу данных для хранения информации о пользователях, заказах, товарах и других сущностях, связанных с учетом продаж. Система должна обеспечивать безопасность пользовательских данных, включая защиту. Эти пункты относятся к техническим требованиям.

## **Обоснование выбора компонентов и технологий для реализации программного средства**

Для реализации программного средства фитнес-центра необходимо выбрать компоненты и технологии, которые обеспечат эффективную и надежную работу системы.

Node.js – это среда выполнения JavaScript, которая позволяет запускать JavaScript-код на сервере. Он основан на движке V8 от Google и обеспечивает масштабируемость и производительность в обработке запросов. Node.js позволяет разрабатывать серверные приложения, веб-серверы и API.

Express является лидером среди базовых JavaScript – фреймворков. Это гибкая среда на базе Node.js для разработки веб и мобильных приложений. Express также можно назвать «фреймворком для других фреймворков», поскольку существует множество других фреймворков, построенных с использованием Express. Ключевые особенности Express.js:

− предлагает надежный механизм маршрутизации для обработки динамических URL-адресов;

− позволяет добавлять дополнительное программное обеспечение в любую точку конвейера обработки запросов;

− упрощает отладку кода.

HTML (HyperText Markup Language) – это язык разметки, используемый для создания структуры и содержимого веб-страниц. С помощью HTML-тегов определяются элементы, такие как заголовки, параграфы, изображения, ссылки и другие, которые определяют визуальное представление и иерархию содержимого страницы.

CSS (Cascading Style Sheets) – это язык стилей, который используется для задания внешнего вида и оформления веб-страниц, созданных с помощью HTML. CSS позволяет определить цвета, шрифты, расположение элементов, размеры, анимации и другие аспекты, которые влияют на визуальное представление страницы.

NPM (Node Package Manager) – это менеджер пакетов для разработки на платформе Node.js. NPM позволяет управлять зависимостями в проекте, устанавливать и обновлять пакеты и библиотеки, которые требуются для разработки программного средства. Он также предоставляет доступ к огромному репозиторию пакетов, где можно найти готовые решения и инструменты для упрощения разработки.

React.js – это JavaScript библиотека, разработанная компанией Facebook, которая используется для создания пользовательских интерфейсов (UI) в веб-приложениях. Она позволяет разрабатывать масштабируемые и динамические веб-приложения с использованием компонентного подхода. React обеспечивает эффективное обновление пользовательского интерфейса, основанного на изменениях данных, без перезагрузки всей страницы.

MongoDB – это кросс-платформенная база данных с открытым исходным кодом, ориентированная на документы и разработанная на C ++, которая является одной из самых популярных и используемых баз данных типа NoSQL. Он работает поверх JSON-подобных документов с парами ключ-значение, схема которых может оставаться неопределенной в каждом документе. MongoDB хранит данные в JSON-подобных документах (называемых BSON), которые могут иметь динамическую схему для документов в одной коллекции. Структуру документа в той же коллекции можно изменить, просто добавив новые поля или удалив существующие.

Все эти компоненты имеют те преимущества, которые лучше всего соответствуют требованиям и ограничениям курсового проекта.

## **Архитектурные решения**

В данной главе будут рассмотрены архитектурные решения для разработки клиент-серверного приложения с использованием model-view-controller (MVC). Архитектура MVC предоставляет структуру и организацию для разделения логики приложения, представления данных и пользовательского интерфейса. Разбивая приложение на эти три компонента, мы повышаем его модульность, упрощаем сопровождение и улучшаем повторное использование кода.

Архитектура MVC состоит из трех основных компонентов: модель, представление и контроллер.

Модель отвечает за обработку данных и бизнес-логику приложения. Здесь выполняются операции чтения и записи данных, взаимодействие с базой данных и другими сервисами.

Представление отображает данные пользователю и обрабатывает взаимодействие с пользователем. В этом компоненте создается пользовательский интерфейс, который предоставляет информацию и возможности для взаимодействия с данными.

Контроллер отвечает за обработку пользовательских действий и управление потоком данных между моделью и представлением. Он получает запросы от представления, вызывает соответствующие операции модели и обновляет представление с новыми данными.

При разработке клиент-серверного приложения на базе архитектуры MVC, компоненты будут распределены между клиентской и серверной сторонами.

К клиентской стороне будет относится представление и контроллер. В клиентской части приложения представление отвечает за отображение данных пользователю и обработку пользовательских действий. Здесь используются HTML, CSS и JavaScript для создания интерфейса пользователя и взаимодействия с сервером через API. А контроллер на клиентской стороне управляет пользовательским взаимодействием и отправкой запросов к серверу. Он обрабатывает пользовательские события, вызывает соответствующие API-запросы и обновляет представление с полученными данными.

К серверной стороне будет относится модель и также контроллер. На серверной стороне модель отвечает за обработку бизнес-логики и взаимодействие с базой данных. Она выполняет операции чтения, записи и обновления данных, обеспечивает безопасность и целостность данных, а также реализует бизнес-правила приложения. Контроллер на серверной стороне принимает запросы от клиента, обрабатывает их, взаимодействует с моделью для получения или обновления данных и возвращает результаты обратно клиенту в виде ответа. Он также может обрабатывать аутентификацию, авторизацию и другую логику безопасности.

Использование Node.js в качестве серверной платформы позволяет разрабатывать быстрые и масштабируемые серверные приложения. Node.js обладает высокой производительностью благодаря своей асинхронной и событийно-ориентированной архитектуре, что особенно важно при обработке большого количества одновременных запросов.

Интеграция MongoDB как базы данных в архитектуре MVC предлагает гибкое хранение и организацию данных. Благодаря документоориентированной модели данных MongoDB, модель приложения может быть разработана с учетом гибких схем данных и динамической структуры. Это особенно полезно при разработке приложений, где требования к данным могут меняться со временем.

Архитектура MVC предоставляет структуру и организацию для разработки клиент-серверных приложений. Разделение на модель, представление и контроллер обеспечивает четкую ответственность каждого компонента и упрощает разработку, тестирование и поддержку приложения. Использование Node.js в качестве серверной платформы и MongoDB в качестве базы данных позволяет создавать быстрые, масштабируемые и гибкие приложения.

## **Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

Схема алгоритма (Flowchart) – это графическое представление последовательности действий или алгоритма, состоящее из различных блоков, стрелок и символов, которые описывают шаги и связи между ними. Она используется для визуализации и описания логики выполнения алгоритма или процесса.

Схемы алгоритмов позволяют наглядно представить последовательность шагов в алгоритме или процессе. Они помогают визуально разбить сложную задачу на более простые шаги и понять логику и порядок выполнения.

На рисунке 2.1 представлен алгоритм покупки тарифа в фитнес-центра.



Рисунок 2.1 – Алгоритм покупки тарифа

На рисунке 2.2 показана вторая схема, реализующая регистрацию.

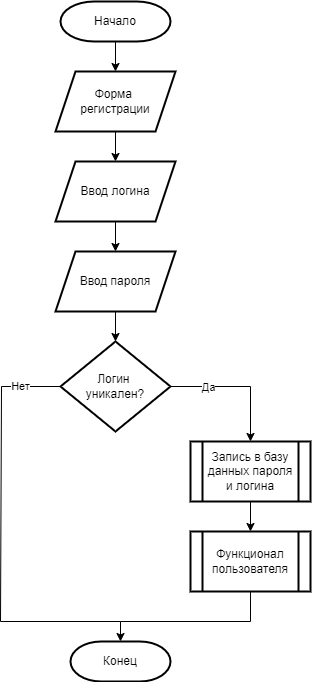


Рисунок 2.2 – Алгоритм регистрации

На рисунке 2.3 представлена схема авторизации.

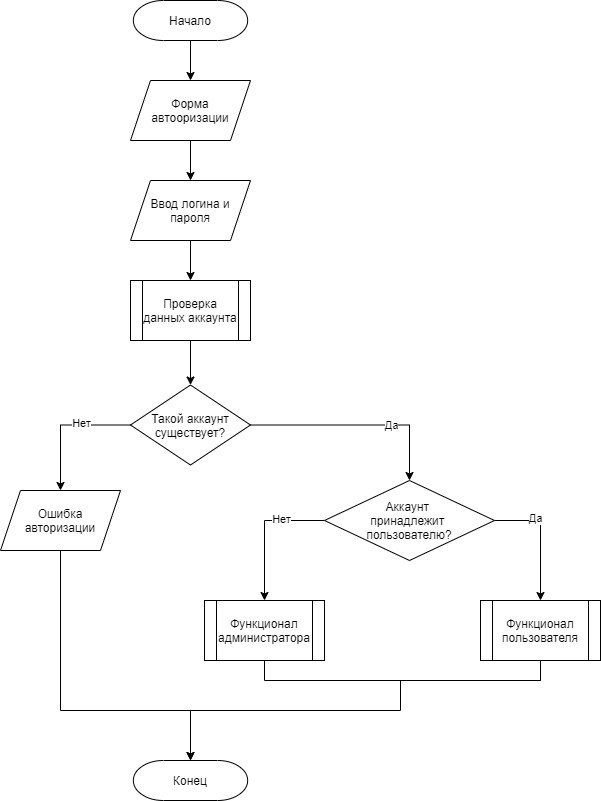


Рисунок 2.3 – Алгоритм авторизации

## **Проектирование пользовательского интерфейса**

Пользовательский интерфейс (UI) – это важный аспект любого программного продукта, в том числе и веб-приложения. Он обеспечивает взаимодействие пользователя с системой и позволяет ему выполнять необходимые действия. Интерфейс помогает двум объектам понимать друг друга и обмениваться информацией. Любой интерфейс в первую очередь должен быть понятным и способным удерживать внимание пользователя.

У веб-сайта, который мы разрабатываем, имеется структура, состоящая из трех основных компонентов: header, основной части сайта и footer.

Header представляет собой верхнюю часть страницы, содержащую меню навигации, которое обеспечивает пользователю доступ к различным разделам и функциональности сайта. Он является важным элементом дизайна и позволяет пользователям быстро ориентироваться и перемещаться по сайту. Она представлена на рисунке 2.4.

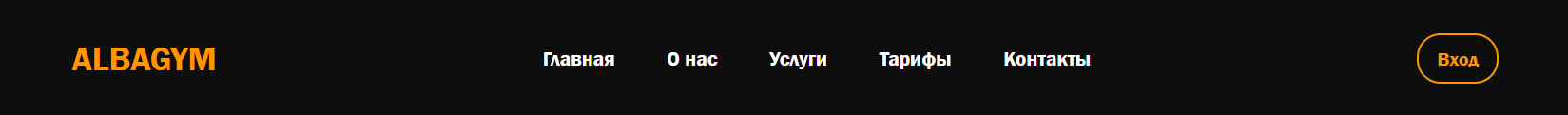


Рисунок 2.4 – Компонент header

Основная часть сайта содержит основной контент и информацию, представленную в виде текста, изображений, видео или других медиаэлементов. Здесь располагается основная информация, предназначенная для ознакомления пользователей с нашим фитнес-центром. Один из примеров представлен на рисунке 2.5.

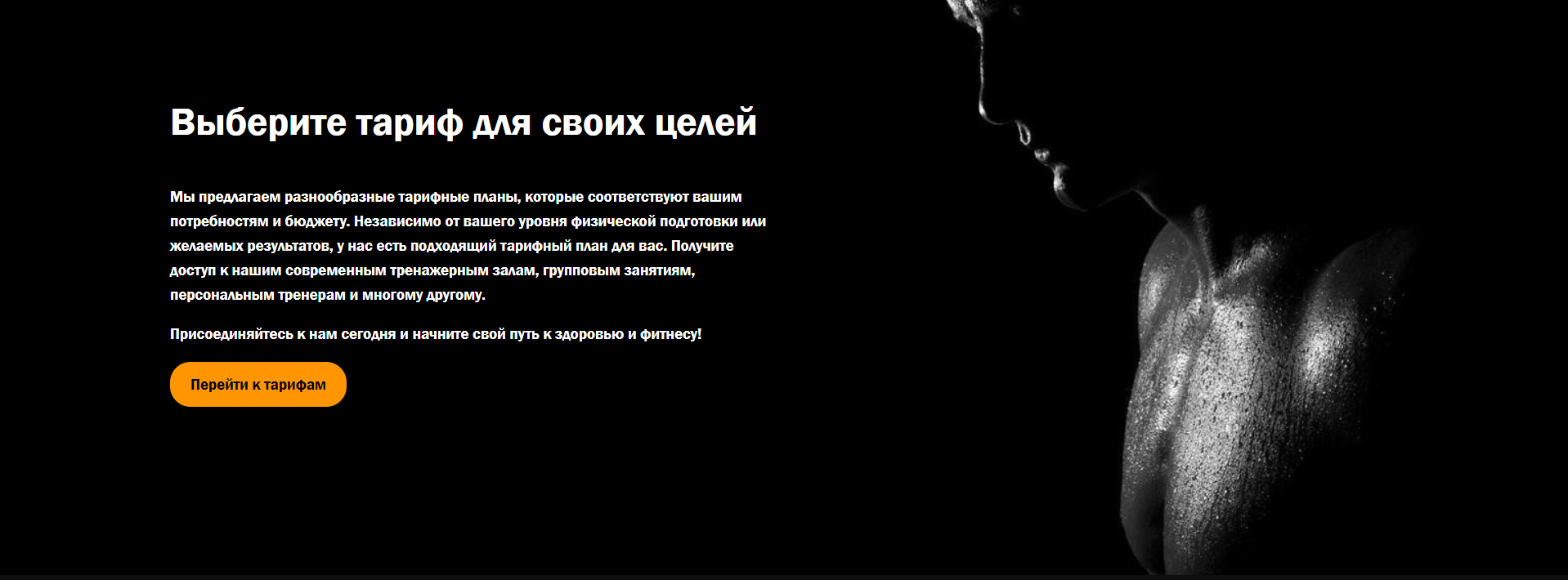


Рисунок 2.5 – Один из примеров основной части

Footer сайта находится в нижней части страницы и содержит место расположения фитнес-центра. На рисунке 2.6 изображен компонент footer.

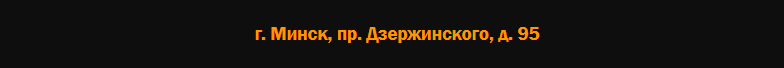


Рисунок 2.6 – Компонент footer

В целом, разработка пользовательского интерфейса является важной частью процесса разработки любого веб-приложения и требует внимательного проектирования и реализации. Описывая процесс разработки пользовательского интерфейса в курсовой работе, можно продемонстрировать понимание важности этого аспекта при создании программного продукта.

## **Методы и средства, используемые для обеспечения безопасности данных**

Для обеспечения безопасности системы применяем несколько методов: шифрование данных, проверка данных на сервере и ограниченный доступ к базе данных.

Для первого метода можно использовать современные алгоритмы шифрования, такие как AES и RSA, для защиты конфиденциальности информации при передаче и хранении данных между клиентскими устройствами и сервером.

К проверке данных на сервере осуществляется через проверку достоверности и целостности данных на серверной стороне, чтобы предотвратить внедрение вредоносного кода или изменение данных на клиентской стороне.

А ограничений доступ к базе данных только с сервера, что помогает предотвратить несанкционированный доступ к хранимым данным.

Эти методы обеспечивают более высокий уровень безопасности системы, минимизируя риски несанкционированного доступа, подделки данных и утечки конфиденциальной информации.

Кроме того, мы реализовали механизмы аутентификации и авторизации, используя безопасные протоколы и хеширования паролей для защиты от несанкционированного доступа, и присваиваем соответствующие права доступа пользователю после успешной аутентификации.

# **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**