Утверждаю

Руководитель предприятия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, печать предприятия)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40-05-01 «Информационные системы и технологии»

ОТЧЕТ

по производственной преддипломной практике

в ООО «Фабрика инноваций и решений» с 10 февраля 2025 г. ­по 21 марта 2025 г.

(наименование предприятия, сроки практики)

Исполнитель

студент 4 курса 1 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Халалеенко А.Н.

(подпись, дата) (Ф.И.О.)

Руководитель практики от предприятия

Кисляк Д.С.

(должность, печать предприятия) (подпись, дата) (Ф.И.О.)

Руководитель практики от университета

Тимонович Г.Л.

(должность, уч. звание) (подпись, дата) (Ф.И.О.)

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2025

**Оглавление**

[Введение 3](#_Toc194134761)

[1 Постановка задачи и аналитический обзор аналогов 4](#_Toc194134762)

[1.1 Описание предметной области 4](#_Toc194134763)

[1.2 Обзор аналогов 4](#_Toc194134764)

[1.2.1 Веб-приложение «silvers screen» 5](#_Toc194134765)

[1.2.2 Веб-приложение «bycard» 6](#_Toc194134766)

[1.2.3 Веб-приложение «afisha.relax.by» 7](#_Toc194134767)

[1.2.4 Анализ по обзору аналогов 9](#_Toc194134768)

[1.3 Постановка задачи и функциональные требования 9](#_Toc194134769)

[1.4 Выводы по разделу 10](#_Toc194134770)

[2 Проектирование веб-приложения 12](#_Toc194134771)

[2.1 Обзор средств разработки 12](#_Toc194134772)

[2.1.1 Технология Nest.js 14](#_Toc194134773)

[2.1.2 Технология TypeORM 15](#_Toc194134774)

[2.1.3 Библиотека React 16](#_Toc194134775)

[2.1.4 База данных PostgreSQL 16](#_Toc194134776)

[2.1.5 Язык SQL 17](#_Toc194134777)

[2.2 Архитектура приложения 18](#_Toc194134778)

[2.3 Проектирование серверной части 19](#_Toc194134779)

[2.4 Проектирование клиентской части 20](#_Toc194134780)

[2.5 Пользовательские роли 20](#_Toc194134781)

[2.6 Проектирование базы данных 22](#_Toc194134782)

[2.7 Выводы по разделу 26](#_Toc194134783)

Введение

Преддипломная производственная практика является ключевым этапом в формировании профессиональных навыков будущего инженера-программиста. В сфере информационных технологий приобретение коммерческого опыта имеет решающее значение для успешного профессионального развития. Практика предоставляет возможность закрепить теоретические знания и применить их в реальных рабочих условиях, что способствует лучшему пониманию особенностей отрасли и ускоренной интеграции в бизнес-среду. Этот этап подготовки служит основой для успешной реализации дипломного проекта, а также помогает быстрее адаптироваться к требованиям индустрии и начать карьеру с уверенностью.

Срок прохождения производственной преддипломной практики с 10 февраля 2025 года по 21 марта 2025 года, а место прохождения — Innowise, компании, специализирующейся на разработке и внедрении IT-решений для различных отраслей. Компания занимается созданием современных решений в области облачных технологий, искусственного интеллекта, машинного обучения, а также предоставляет услуги по разработке программного обеспечения и консалтингу для бизнеса. Юридический адрес компании — город Витебск, проспект Строителей, 11А, офис 209.

Innowise разработала всестороннюю программу производственной практики, которая направлена на глубокое погружение в процессы разработки программного обеспечения. Программа включает в себя знакомство с основными этапами жизненного цикла продукта, особенностями работы IT-компаний, а также внутренними процессами, что позволяет студенту быстро адаптироваться к реальной рабочей среде. Благодаря комплексному подходу, включая практические задания, семинары и наставничество опытных разработчиков, участник практики сможет освоить современные методы разработки, которые существенно помогут в выполнении дипломного проекта и обеспечат успешное начало профессиональной карьеры.

1. Постановка задачи и аналитический обзор аналогов

**1.1 Описание предметной области**

Предоставление услуг кинотеатра включает в себя не только показ фильмов, но и создание удобства для посетителей, начиная от процесса бронирования билетов до момента просмотра фильма. В современном мире, где цифровые технологии играют ключевую роль во всех сферах жизни, кинотеатры также адаптируются к новым стандартам обслуживания клиентов.

Бронирование билетов через интернет или мобильное приложение становится все более популярным способом приобретения билетов. Это позволяет пользователям выбрать удобное для них время и место просмотра, а также избежать очередей в кассах кинотеатра. Технологии мониторинга и анализа данных помогают кинотеатрам лучше понимать предпочтения своих клиентов, что в свою очередь способствует улучшению качества обслуживания и предложений.

Кинотеатры, использующие современные IT-системы, могут предлагать клиентам широкий спектр услуг, включая онлайн-бронирование мест, выбор места в зале, оплату билетов с использованием банковских карт или электронных кошельков, получение персонализированных рекомендаций по фильмам. Эти возможности делают процесс посещения кинотеатра более комфортным для пользователей.

Администраторы систем управления кинотеатрами отвечают за обеспечение бесперебойной работы всех сервисов, от обработки платежей до поддержания актуальности информации о прокате. Использование специализированных программных решений позволяет им анализировать данные о посещаемости, предпочтениях аудитории и эффективности маркетинговых кампаний, что в свою очередь помогает принимать обоснованные решения по развитию бизнеса.

Комплексные системы управления кинотеатрами включают в себя не только функциональность бронирования билетов, но и инструменты для управления кон тентом (фильмами), планирования расписания, учета доходов и расходов, а также анализ эффективности работы кинотеатра. Такие системы позволяют кинотеатрам быть конкурентоспособными на рынке развлечений, предлагая своим клиентам высококачественный сервис и интересный контент. Все вышеперечисленное делает сферу администрирования кинотеатра и покупки билетов онлайн довольно перспективной и востребованной в современных реалиях.

1.2 Обзор аналогов

Для создания принципиально нового решения в виде программного продукта, который будет использоваться для просмотра и заказа билетов на фильмы, важно провести тщательный анализ существующих программных средств в данной сфере.

Данный анализ должен охватывать как достоинства, так и недостатки текущих продуктов и их аналогов, чтобы на основе полученной информации, выделенных преимуществ и недостатков, сформулировать четкие требования к проектируемому программному средству. Основная цель такого подхода заключается в том, чтобы учесть опыт успешных и менее удачных разработок, а также внести в них улучшения или внесения изменений, которые сделают новый продукт более привлекательным и удобным для пользователей.

При выборе программных продуктов для анализа в качестве исследуемых аналогов, следует обращать внимание на те, которые наиболее тесно связаны с областью применения будущего программного продукта. Это может включать в себя различные платформы для онлайн-просмотра фильмов, сервисы для бронирования билетов в кинотеатрах, а также приложения для мобильных устройств, предлагающие подобного рода услуги.

Источниками информации для анализа могут служить электронные базы данных, доступные в сети Интернета, а также отзывы пользователей и профессиональные обзоры в специализированных изданиях.

Процесс анализа начинается с изучения функциональности и интерфейса каждого из исследуемых продуктов, чтобы определить, какие элементы могут быть полезны для нового проекта, а какие требуют доработки или замены. Особое внимание уделяется вопросам безопасности и удобства использования, поскольку эти аспекты напрямую влияют на восприятие продукта потенциальными пользователями.

На основе собранной информации формируется список требований к проекту, включающий в себя функциональные, технические и пользовательские требования. Этот список затем используется в процессе разработки приложения, обеспечивая его соответствие ожиданиям целевой аудитории и конкурентоспособность на рынке.

Таким образом, проведение детального анализа существующих программных средств позволяет не только избежать повторения ошибок прошлых разработок, но и создать продукт, который будет отвечать потребностям и ожиданиям пользователей, предлагая им новые возможности и улучшенные условия для просмотра и заказа билетов на фильмы.

1.2.1 Веб-приложение «silvers screen»

Веб-приложение по покупки билетов silverscreen.by. Сервис не особо по пулярен из-за слабой рекламы и продвижения, вы даже можете не найти его на первых страницах поиска (данный сервис служит только для покупок билетов в кинотеатры silverscreen), но меня привлек этот сервис максимальной функциональностью и красотой дизайна. На данном сайте достаточно минималистичный дизайн, в тоже время, привлекающий внимание пользователей.

Также предоставляется возможность оставить отзывы о фильмах и кинотеатрах, поделиться своим мнением с другими пользователями. Выбор места в зале сопровождается выводом стоимости этого места, что довольно удобно. Интер фейс данного сервиса представлен на рисунке 1.1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.1 – Интерфейс silverscreen.by

Также сервис предлагает пользователям не только детальные инструкции, но и широкий спектр категорий мест для удобства и персонализированного подхода к каждому пользователю. Этот подход показывает глубину понимания потребностей пользователей, помогая им находить идеальное решение для каждого, по его средствам и запросам. Удачный сервис в плане реализации, функционала и дизайна. Из минусов только малая известность.

1.2.2 Веб-приложение «bycard»

Веб-приложение по покупки билетов bycard.by – один из самых популярных и востребованных сайтов по продажи билетов на любые мероприятия, проходящие на территории Беларуси, на нем можно найти интересующие вас спектакли, фильмы, другие события и купить/забронировать билеты. Интерфейс представлен на рисунке 1.2.

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.2 – Интерфейс Bycard.by

Функционал сервиса включает в себя:

* выбор досуга;
* выбор мест проведения мероприятий;
* выбор организаторов;
* фильтры;
* поисковую систему;
* выбор региона;
* разбивка по мероприятиям.

Отличный сервис, функционал реализован на 100%, есть все нужные сервисы, недостатки крайне малы и нишевы. Например, сам по себе сервис берет процент с каждого проданного билета, за счет чего не все кинотеатры хотят сотрудничать с ним, а также стоимость покупки билета онлайн повышается. Еще небольшим минусом можно считать, что при возврате билета деньги будут возвращены только через несколько дней, но это скорее минус системы оплаты, а не самого сайта.

1.2.3 Веб-приложение «afisha.relax.by»

Это веб-приложение представляет собой удобный инструмент для тех, кто ищет новые возможности для отдыха и развлечений. Он предлагает широкий спектр мероприятий, от культурных событий до спортивных соревнований, делая выбор наиболее подходящего варианта максимально простым и быстрым. Возможности для пользователей разнообразны, начиная от просмотра каталога мероприятий и заканчивая прямой регистрацией на интересующие вас события. Главная страница представлена на рисунке 1.3.

Изображение выглядит как текст, снимок экрана, Веб-сайт, Человеческое лицо

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.3 – Главная страница веб-приложения «afisha.relax.by»

Одним из ярких преимуществ этого сервиса является особое внимание, уделяемое группам студентов и пенсионеров. Предоставление скидок этим категориям пользователей подчеркивает социальную ответственность сервиса и делает его более доступным для широкого круга людей.

Скидки для студентов и пенсионеров являются важной составляющей стратегии привлечения новых пользователей и укрепления лояльности существующих. Для студентов это возможность экономить на развлечениях и мероприятиях, что особенно актуально во время учебы, когда бюджет часто ограничен. Для пенсионеров же такие скидки позволяют участвовать в культурных и развлекательных мероприятиях без значительных финансовых затрат, что способствует их активному отдыху и социализации. На рисунке 1.4 представлен пример скидок.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.4 – Пример скидок в веб-приложении afisha.relax.by

В целом, сервис похож на предыдущий, с похожим функционалом, но более социализированный и направленный на все группы населения. Сервис является эволюционным продолжением предыдущего, но с более широким спектром функциональных возможностей и ориентацией на все слои населения. Кроме того, сервис имеет более простой и интуитивно понятный интерфейс, что делает его доступным для пользователей с любым уровнем технической грамотности.

1.2.4 Анализ по обзору аналогов

Проанализировав данные сервисы, я обнаружил, что общие характеристики это: возможность оформления заказа, поиск фильмов по названию, разбиение филь мов по жанрам, но также ни у одного из них нельзя добавить фильмы во вкладку «Любимые», что я считаю существенным минусом.

На основе проведенного анализа можно выделить ключевые характеристики, которые следует учесть при разработке нового приложения для покупки билетов на фильмы. Важными элементами приложения должны быть удачная реализация, функциональность и привлекательный дизайн, а также эффективные меры по рекламе и продвижению для повышения его известности. Особое внимание следует уделить широкому функционалу, включая возможность выбора досуга, сохранения понравившихся фильмов, а также удобная поисковой системы.

Кроме того, важными аспектами являются возможность оформления заказа, поиск фильмов и функция разбиения фильмов по жанрам.

Добавление в приложение вкладки «Любимые» для сохранения интересных фильмов пользователем также представляется важным дополнением, которое может повысить удовлетворенность пользователей.

Исключение недостатков, таких как комиссии с продажи билетов и повышение стоимости при онлайн-покупке, а также предоставление удобных механизмов возврата билетов, будет способствовать улучшению общего восприятия приложения пользователями. Исходя из этого был сформирован функционал приложения:

* поиск фильмов;
* возврат билетов;
* изменение личных данных;
* выбор одного или нескольких мест;
* покупка билетов;
* просмотр информации о фильмах;
* добавление фильма в избранные;
* просмотр жанров;
* просмотр статистики по фильмам;
* добавление фильмов/сеансов/залов;
* удаление фильмов/сеансов;
* блокировка/разблокировка пользователей.

1.3 Постановка задачи и функциональные требования

Оценка аналогичных решений и их возможностей позволяет определить набор функциональных требований для разрабатываемой системы. Функциональные требования описывают те функциональные возможности программного обеспечения, которые разработчики должны реализовать, чтобы пользователи смогли выполнять свои задачи в соответствии с бизнес-целями.

Бизнес-требования определяют высокоуровневые цели организации или заказчиков системы. Как правило, они формулируются теми, кто финансирует проект, заказывает систему, отвечает за работу пользователей, отвечает за маркетинг.

Дипломный проект не имеет конкретного заказчика, способного сформулировать бизнес-требования, поэтому в качестве таких высокоуровневых требований можно использовать общие требования к разрабатываемому решению. Функциональные требования могут быть описаны в виде заявлений о необходимых возможностях, способах взаимодействия и методах реализации.

Основной целью приложения является разработка оптимальной системы с интуитивно понятный интерфейс для взаимодействия администрации кинотеатра с клиентами, со всем необходимым функционалом, достаточным и полноценным для управления кинотеатрам, а также понятным и доступным для клиента. На основе поставленной цели можно сформировать основной функционал системы:

* приложение должно поддерживать роли: гость, администратор, пользователь;
* к общим функциям авторизованных пользователей относятся: изменение данных профиля и пароля, просмотр доступных сеансов на фильмы, покупка билетов, добавление фильмов в «Любимые», просмотр фильмов по категориям, про смотр своих билетов, поиск фильмов;
* к функциям администратора относится: просмотр, создание, удаление фильмов/сеансов, так же добавление залов, блокировка/разблокировка пользователей, про смотр статистики билетов по фильмам, проверка билетов пользователя перед сеансом.

На основе анализа аналогов сформулированы следующие в пользу разработки веб-приложения для организации продажи билетов в кинотеатр:

* на данный момент есть несколько хороших приложений, но каждый со своими недостатками;
* в будущем разрабатываемое приложение имеет большой потенциал развития и интеграции с другими системами;
* на данный момент нет достаточного количества качественных аналогов;
* популярность покупки билетов онлайн растет с каждым годом;
* в условиях быстрого развития информационных технологий веб-приложение для администрирования кинотеатра является актуальным и востребованным для многих кинотеатров, что на данный момент пользуются чужими системами.

Таким образом, решение о разработке веб-приложения для бронирования междугородних пассажирских перевозок является обоснованным и рациональным.

1.4 Выводы по разделу

В разделе представляем теоретические основы, описывающие предметную область, а также результаты поиска и анализа аналогичных веб-приложений. Данный анализ позволил нам выявить преимущества и недостатки существующих решений, чтобы при разработке собственного приложения избежать схожих ошибок.

В частности, мы провели исследование следующих аспектов:

* основы предметной области, включая понятия, модели и принципы, лежащие в основе разработки веб-приложений;
* анализ существующих решений, включая их функциональные возможности, преимущества и недостатки;
* определение функциональных требований к разрабатываемому приложению, включая требования к интерфейсу, функциональности и производительности.

Результаты анализа и поиска помогли нам сформулировать функциональные требования к разрабатываемому приложению, которые будут учитываться при его разработке. Это позволит создать приложение, которое будет отвечать потребно стям пользователей и иметь конкурентные преимущества на рынке.

2 Проектирование веб-приложения

Процесс проектирования архитектуры программного обеспечения состоит в проектировании структуры всех его компонент, функционально связанных с решаемой задачей, включая сопряжения между ними и требования к ним.

Архитектура программного обеспечения в традиционном смысле включает определение всех модулей программ, их иерархии и сопряжения между ними и данными. Веб-приложение – это решение, в основе которого лежит взаимодействие браузера и веб-сервера. Такие приложения являются кроссплатформенными сервисами, доступными с любого современного устройства, и не привязаны к архитектуре сети.

Этап проектирования веб-приложения можно логически разбить на три основополагающие части:

* проектирование базы данных;
* проектирование серверной части;
* проектирование клиентской части.

Использование приложения начинается с пользовательского интерфейса, который реализован при помощи React. Пользователь выполняет действие, посылает запрос по адресу развернутого на сервере NestJS веб-API приложения. В свою очередь управление переходит к сервисам NestJS и модели. Если запрос требует работы с данными, используется TypeORM. PostgreSQL выдает необходимые данные. TypeORM передает эти данные веб-API, который формирует ответ пользователю.

Важно отметить, что подобная архитектура является классическим примером «N-layer» архитектуры, где каждый слой зависит только от предыдущего компонента. Существует множество различных видов и типов архитектур, которые успешно применяются. Одной их наиболее используемых является классическая трехуровневая система, которая подразумевает разделение приложения на три логических уровня. Следует отметить, что многоуровневой архитектурой часто обозначают два не совсем связанных понятия: n-layer и n-tier. И layer, и tier, как правило, обозначаются словом «Уровень», иногда по отношению к «Layer» еще употребляется слово «Слой». Однако в обоих случаях уровни будут разного порядка. Также взаимодействие происходит последовательно. Так, к примеру, последний компонент не способен взаимодействовать с первым.

Всего выделяют три абстрактных уровня: уровень представления, необходимый для работы пользователя (располагаются интерфейсы, которые пользователь видит и с которыми взаимодействует), уровень бизнес-логики, где осуществляется основная обработка данных, и уровень доступа к данным, необходимый для их хранения и использования. При этом следует отметить, что крайние уровни не могут взаимодействовать между собой, то есть уровень представления не может напрямую обращаться к базе данных и даже к уровню доступа к данным, а только через уровень бизнес-логики.

2.1 Обзор средств разработки

JavaScript (JS) – это высокоуровневый, динамический и интерпретируемый язык программирования, который в основном используется для создания веб-приложений и добавления интерактивных элементов на веб-сайты. Он был создан в 1995 году Бренданом Эйхом, когда он работал в Netscape Communications Corporation.

Одна из ключевых особенностей JavaScript – его способность выполняться на стороне клиента в браузере, что позволяет создавать динамические и интерактивные веб-страницы. Это означает, что JavaScript может реагировать на ввод пользователя в реальном времени, что делает его идеальным языком для создания веб-приложений, требующих взаимодействия с пользователем.

JavaScript также является универсальным языком, который может использоваться для широкого спектра применений за пределами разработки веб-приложений. Напри мер, он может использоваться для серверной разработки с помощью Node.js, разработки игр с помощью фреймворков, таких как Phaser, и даже для разработки мобильных и десктопных приложений с помощью фреймворков, таких как Electron.

Другая важная особенность JavaScript – его способность работать с асинхронным кодом. Это означает, что JavaScript может обрабатывать несколько задач одно временно без блокировки выполнения других операций. Это особенно полезно для создания высокопроизводительных веб-приложений, требующих быстрой реакции.

JavaScript также является прототипным языком, что означает, что он использует прототипы вместо классов для создания объектов. Это позволяет использовать более гибкую и динамическую объектно-ориентированную парадигму программирования, поскольку объекты могут получать свойства и методы от других объектов во время выполнения.

В синтаксисе JavaScript является языком в стиле C, похожим на языки, такие как Java и C++. Он поддерживает широкий спектр парадигм программирования, включая процедурную, объектно-ориентированную и функциональную парадигмы.

JavaScript является популярным языком, имеющим большую и активную со общество разработчиков. Это сообщество создало широкий спектр библиотек, фреймворков и инструментов, которые облегчают разработку сложных веб-приложений. Некоторые из самых популярных фреймворков и библиотек JavaScript включают React, Angular, Vue.js и jQuery.

Вкратце, JavaScript – мощный и универсальный язык программирования, который в основном используется для создания веб-приложений и добавления интерактивных элементов на веб-сайты. Его способность выполняться на стороне клиента в браузере, работать с асинхронным кодом и использовать прототипы для объектно-ориентированного программирования делают его идеальным языком для создания высокопроизводительных веб-приложений, требующих быстрой реакции.

Кроме того, его большое и активное сообщество разработчиков создало широкий спектр библиотек, фреймворков и инструментов, которые облегчают разработку сложных веб-приложений. Благодаря этим ресурсам разработчики могут использовать готовые решения для множества задач, таких как аутентификация пользователей, обработка данных, интеграция с базами данных, работа с API, управление состоянием приложений и многое другое.

2.1.1 Технология Nest.js

Nest.js является одним из самых популярных фреймворков для создания серверных приложений на основе Node.js. Он построен на экосистеме Express.js и предоставляет ряд дополнительных возможностей, таких как модульная структура приложения, внедрение зависимостей, декораторы и другие функции, которые де лают разработку приложений более простой и эффективной.

Nest.js использует декораторы для определения контроллеров и провайдеров. Контроллеры отвечают за обработку входящих запросов, а провайдеры предоставляют логику приложения. Nest.js также поддерживает использование различных баз данных, таких как MongoDB, PostgreSQL, MySQL и других.

Одним из ключевых преимуществ Nest.js является его совместимость с Angular, фреймворком для клиентской разработки от Google. Angular и Nest.js ис пользуют общий язык разработки TypeScript, что позволяет легко интегрировать клиентскую и серверную части приложения.

Nest.js также предоставляет поддержку микросервисной архитектуры, что позволяет создавать сложные приложения, состоящие из множества взаимосвязанных сервисов. Кроме того, Nest.js имеет активную и растущую сообщество разработчиков, что обеспечивает доступность множества сторонних модулей и библиотек.

В целом, Nest.js является мощным и гибким фреймворком для разработки серверных приложений на основе Node.js. Его функциональные возможности, совместимость с Angular и активная сообщество делают его одним из лучших вы боров для разработки сложных веб-приложений. Основное взаимодействие про исходит по протоколу HTTP.

При вызове операций с контроллера важный момент играет routing. Routing позволяет подсистеме WebApi связать Uri адрес и конкретную операцию из контроллера.

С помощью NestJS можно легко создавать REST-сервисы. До него создание сервисов было более сложным, и эта технология предоставляет широкие возможности хостинга и позволяет выполнять следующие задачи:

* создавать веб-приложения и службы;
* использовать избранные средства разработки в Windows, macOS и Linux;
* выполнять развертывания в облаке или локальной среде;
* запускать в Node.js.

NestJS предоставляет следующие преимущества:

* единое решение для создания пользовательского веб-API;
* разработанное приложение удобно тестируется;
* модульная архитектура приложения;
* поддержка внедрения зависимостей;
* возможность использования декораторов для контроллеров и провайдеров;
* поддержка баз данных, таких как MongoDB, PostgreSQL, MySQL;
* совместимость с фреймворком Angular;
* активное и растущее сообщество разработчиков.

NestJS проект предоставляет следующие возможности размещения:

* Node.js;
* Docker.

Кроме того, NestJS также может быть развернут на различных cloud-платфор мах, таких как:

* Heroku;
* DigitalOcean;
* AWS Elastic Beanstalk;
* Google App Engine;
* Railway;
* Digital Ocean App Platform.

Каждая из этих платформ имеет свои преимущества и недостатки, и выбор за висит от конкретных требований проекта.

Например, AWS Elastic Beanstalk − это управляемый сервис, который позво ляет быстро развернуть и управлять приложениями в облаке AWS без необходимо сти беспокоиться об инфраструктуре, которая запускает эти приложения. Digital Ocean App Platform − это сервис PaaS, который позволяет быстро создавать, развер тывать и масштабировать приложения и статические сайты. Railway − это платформа, которая позволяет легко развернуть приложение в облаке за несколько ми нут, поддерживая множество языков программирования, включая Java, PHP, Deno, Python, Swift, Go и другие.

2.1.2 Технология TypeORM

TypeORM является объектно-реляционной картой (ORM) для TypeScript и JavaScript, которая облегчает взаимодействие с базами данных с помощью объектно-ориентированного синтаксиса. Это может упростить доступ к базе данных и улучшить организацию кода в дипломном проекте.

Одним из преимуществ использования TypeORM является его поддержка многих баз данных, таких как PostgreSQL, MySQL, MariaDB, SQLite, Microsoft SQL Server и других. Это означает, что вы можете выбрать базу данных, которая наилучшим образом подходит для вашего проекта, и TypeORM обеспечит согласованный интерфейс для работы с ней. Это может сэкономить время и усилия, поскольку вы не будете обучаться новому синтаксису или API для каждой базы данных, с которой вы работаете.

TypeORM также предоставляет мощный набор функций для работы с базами данных, включая поддержку сущностей, репозиториев и отношений. Сущности – это классы, которые сопоставляются с таблицами базы данных, а репозитории предоставляют способ взаимодействия с этими сущностями. Отношения позволяют определять связи между сущностями, облегчая работу со сложными структурами данных.

Другое преимущество использования TypeORM – его интеграция с TypeScript. TypeScript – это статически типизированное надмножество JavaScript, которое до бавляет необязательные типы, классы и модули к языку. Используя TypeScript с TypeORM, вы можете воспользоваться системой типов TypeScript для раннего выявления ошибок и улучшения удобочитаемости кода.

Кратко, TypeORM – это мощная и гибкая ORM, которая может упростить доступ к базе данных и улучшить организацию кода в проекте. Его поддержка многих баз данных, сущностей, репозиториев и отношений делает его 19 отличным выбором для работы со сложными структурами данных, а его интеграция с TypeScript может улучшить качество кода и удобочитаемость.

2.1.3 Библиотека React

React – это мощная библиотека JavaScript для создания пользовательских интерфейсов. Она позволяет создавать повторно используемые компоненты пользовательского интерфейса и эффективно обновлять, и рендерить эти компоненты. React позволяет постепенно вводить его в свой проект, используя только необходимую функциональность в любой момент времени.

React имеет богатую экосистему с многими передовыми функциями и концепциями для исследования, такими как контекст и ссылки. API-документация и глоссарий предоставляют подробную информацию об этих темах, а также ответы на часто задаваемые вопросы. React позволяет создавать переиспользуемые компоненты, которые могут быть использованы в различных частях приложения. Это упрощает разработку и поддержку приложений, поскольку позволяет избежать дублирования кода и повышает модульность.

Компоненты в React представляют собой независимые единицы, которые могут принимать входные данные в виде пропсов и отображать соответствующий пользовательский интерфейс. Эти компоненты могут быть использованы в других компонентах или страницах, что позволяет легко изменять и обновлять пользовательский интерфейс без необходимости изменять основной код.

В целом, использование компонентов в React позволяет создавать легковесные, модульные и поддерживаемые приложения, уменьшая дублирование кода и упрощая разработку.

Во время работы над приложением React помог создать динамичный, удобный для обслуживания и масштабируемый пользовательский интерфейс. Кроме того, React позволяет легко интегрировать сторонние библиотеки и API, что расширяет функциональность приложения без лишних затрат времени. Его компонентная архитектура и эффективный рендеринг делают его отличным выбором для современной веб-разработки. Используя возможности React, вы можете создать отполированный и профессиональный проект, демонстрирующий ваши навыки и компетенции.

2.1.4 База данных PostgreSQL

PostgreSQL является мощной и надежной реляционной базой данных, которая используется в дипломном проекте. Она отличается масштабируемостью, безопасностью и функциональностью, что делает ее идеальным выбором для приложений.

Одна из ключевых особенностей PostgreSQL – это возможность создавать собственные расширения и модули, которые могут быть интегрированы непосредственно в базу данных. Это позволяет нам настраивать и расширять функциональность PostgreSQL в соответствии с нашими потребностями, что особенно полезно в больших и сложных проектах.

Кроме того, PostgreSQL поддерживает множество различных типов данных, включая стандартные типы, такие как целые числа, вещественные числа и строки, а также более сложные типы, такие как JSON, XML и географические данные. Это позволяет нам работать с разнообразными данными в единой системе, что упрощает разработку и поддержку приложений.

В плане производительности PostgreSQL также демонстрирует отличные результаты. База данных может обрабатывать тысячи запросов в секунду, а также поддерживать сотни и даже тысячи одновременных соединений. Кроме того, PostgreSQL поддерживает различные механизмы репликации и отказоустойчивости, которые позволяют нам создавать отказоустойчивые и масштабируемые системы.

Однако, несмотря на все свои преимущества, PostgreSQL также имеет некоторые ограничения. Например, база данных может требовать больше ресурсов, чем другие решения, такие как NoSQL-базы данных. Кроме того, PostgreSQL может быть сложнее в настройке и управлении, особенно для начинающих пользователей.

В целом, PostgreSQL – мощная и функциональная реляционная база данных, которая отлично подходит для многих приложений. Ее возможность настраивать и расширять функциональность, а также поддержка разнообразных типов данных делают ее идеальным выбором для сложных и масштабных проектов. Однако, необходимо учитывать ее ограничения и требования к ресурсам при выборе решения для своего дипломного проекта.

2.1.5 Язык SQL

Общепринятым стандартом языка работы с реляционными базами данных в настоящее время является язык структурированных запросов – SQL. Это универ сальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. Вопреки существующим заблуждениям, SQL является информационно-логическим языком, а не языком программирования.

SQL основывается на реляционной алгебре. Язык SQL делится на три части:

− операторы определения данных;

− операторы манипуляции данными (Insert, Select, Update, Delete);

− операторы определения доступа к данным.

При всех своих изменениях SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных.

Преимущества использования SQL:

* независимость от конкретной СУБД (несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую);
* наличие стандартов (наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует стабилизации языка);
* декларативность (с помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса).

Недостатки:

* сложность. Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов он стал настолько сложным, что превратился в инструмент программиста;
* отступления от стандартов. Многие разработчики СУБД вносят изменения в язык SQL, применяемый в разрабатываемой СУБД. Таким образом появляются специфичные для каждой конкретной СУБД диалекты языка SQL.

2.2 Архитектура приложения

Архитектура программного обеспечения – совокупность важнейших решений об организации программной системы. Основные задачи разработки архитектуры программного средства:

− выделение программных подсистем и отображение на них внешних функций (заданных во внешнем описании) программного средства;

− определение способов взаимодействия между различными выделенными программными подсистемами.

Всего в приложении используется 3 основных сервиса:

– сервис, реализующий клиентскую часть приложения;

– сервис, реализующий серверную часть приложения;

– TypeORM – сервис, реализующий доступ базе данных.

Диаграмма развертывания – это тип диаграммы в UML, которая демонстрирует физическую структуру системы, включая аппаратное обеспечение и программное обеспечение, на котором она работает. Она показывает, как программное обеспечение развертывается на физических устройствах, называемых узлами, и как эти узлы взаимодействуют друг с другом для выполнения функций системы. Диаграмма развертывания помогает визуализировать архитектуру системы, созданную для физической реализации, и моделирует распределение программного обеспечения по физическим узлам в распределенных системах.

Диаграмма развертывания представлена на рисунке 2.1 и в приложении А.

Изображение выглядит как текст, диаграмма, зарисовка, План

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.1 – Диаграмма развертывания

Каждый сервис программного средства изолирован от других сервисов, что позволяет легко расширять функционал приложения. Клиентские запрос изначально обрабатываются NestJs сервером, далее React сервис генерирует представление, которое проходит через маршрутизатор, далее отображается на странице веб-браузера.

2.3 Проектирование серверной части

Для реализации серверной части клиент-серверного приложения был выбран фреймворк NestJS. NestJS – это гибкий фреймворк для создания серверных приложений на платформе Node.js. Он предоставляет удобный набор функций и инструментов, позволяющих разработчикам быстро и эффективно создавать веб–серверы и API. NestJS обладает простым и понятным API, что делает процесс разработки более удобным и интуитивным. Он также предоставляет множество расширений и плагинов, которые облегчают разработку различных функциональностей, таких как маршрутизация, обработка запросов, обработка сессий и многое другое.

Архитектура серверной части была разработана на основе «слоистой архитектуры» с некоторыми изменениями для повышения удобства использования. Слои стая архитектура представляет собой подход к проектированию программного обеспечения, при котором приложение разбивается на логические слои или уровни, каждый из которых выполняет определенные функции.

Слоистая архитектура разделяет систему на несколько уровней, каждый из которых имеет свои задачи и границы взаимодействия. Внешний уровень обрабатывает пользовательский интерфейс, уровень представления преобразует данные, уровень бизнес-логики содержит основную логику приложения, а уровень данных работает с базами данных или другими источниками данных. Такой подход способствует модульности, упрощает разработку и обеспечивает изоляцию различных компонентов системы.

Для связи с базой данных была использована ORM TypeORM. TypeORM – это современный ORM (Object-Relational Mapping), предназначенный для связи приложений с базами данных. ORM является программным инструментом, который позволяет разработчикам работать с данными в базе данных, используя объектно-ориентированный подход, вместо прямой работы с SQL-запросами.

TypeORM предоставляет удобный и выразительный API для выполнения операций чтения, записи, обновления и удаления данных в базе данных. Он позволяет разработчикам взаимодействовать с базой данных, используя язык программирования, с которым они уже знакомы (например, JavaScript или TypeScript), а не прямо работая с SQL-запросами.

2.4 Проектирование клиентской части

Для проектирования клиентской части приложения была выбрана такая библиотека, как React. React – это JavaScript библиотека c открытым исходным кодом для создания пользовательских интерфейсов.

Для визуального оформления и организации компонентов на странице сайта была использована библиотека Mantine UI. Mantine UI – это популярный инструмент для разработки веб-интерфейсов, который предлагает набор предварительно созданных компонентов, стилей и скриптов. Он облегчает и ускоряет процесс создания и поддержки современных веб-страниц, которые легко адаптируются для различных устройств.

2.5 Пользовательские роли

При проектировании приложения, его серверной и клиентской части, а также реализации его функционала была учтена диаграмма вариантов использования, указанная в приложении Б.

В разрабатываемом веб-приложении присутствует 2 роли. В зависимости от той или иной роли определяются функциональные возможности в рамках разрабатываемой системы.

В рамках системы существует администратор, пользователь.

Рассмотрим функциональность, которая предоставляется администратору: данный пользователь имеет определенное количество функций и выполняет работу по поддержанию работоспособности веб-приложения. Ему доступны следующие функции:

* поиск фильмов;
* возврат билетов;
* изменение личных данных;
* покупка билетов; – просмотр фильмов;
* добавление фильма в избранные;
* просмотр истории действий;
* добавление и удаление сеансов\фильмов;
* просмотр статистики по фильмам;
* блокировка\разблокировка пользователей;
* проверка билетов на сеанс по QR-code.

Фрагмент диаграммы вариантов использования администратором приложения представлен на рисунке 2.2.

Изображение выглядит как текст, диаграмма, рисунок, зарисовка

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.2 – Фрагмент диаграммы вариантов использования администратора

У авторизованного пользователя в соответствии и диаграммой должны быть доступен следующий функционал:

– поиск фильмов;

– возврат билетов;

– изменение личных данных;

– покупка билетов;

– просмотр фильмов;

– добавление фильма в избранные.

Описание вышеперечисленных функциональных возможностей пользователя представлено на рисунке 2.3.

Изображение выглядит как текст, диаграмма, круг, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.3 – Фрагмент диаграммы вариантов использования пользователя

Основные функции администратора – это проверка билетов, а основной функцией пользователя – это покупка билетов. Покупка билетов происходит по следующему алгоритму: сначала пользователь выбирает билет или билеты, далее нажимает на кнопку оплатить, после этого его перенаправляет на платежную систему, пользователь вводит реквизиты своей карты и после успешной оплаты его перенаправляет обратно на страницу фильма. Проверка билетов осуществляется с помощью сканера, который использует камеру для получения данных с qr-code, после этого данные билета сравниваются с данными из базы данных и при их совпадении пользователя пропускают на сеанс. Более подробно расписаны алгоритмы проверки билетов и их покупки в Приложении В и Приложении Г.

2.6 Проектирование базы данных

Для хранения данных в приложении было выбрано реляционное базовое хранилище PostgreSQL. PostgreSQL представляет собой систему управления реляционными базами данных, обладающую широким спектром функциональности для работы с данными в структурированном формате.

Выбор PostgreSQL в качестве реляционной базы данных был обусловлен его высокой производительностью и гибкостью настройки. PostgreSQL предоставляет возможности для оптимизации и настройки конфигурации, что позволяет достичь высокой скорости работы в соответствии с требованиями нашего приложения.

Схема базы данных представлена на рисунке 2.4, а также в Приложении Д.

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.4 – Логическая схема базы данных

При проектировании данной базы данных я руководствовался принципами нормализации, первому второму и третьему принципу, что бы в итоге моя база данных полностью соответствовала третьему принципу нормализации.

За счет данных принципов у меня структурированная и эффективная база данных, которая облегчает хранение, доступ и обработку информации о кинотеатре.

Для хранения данных приложение использует базу данных. База данных состоит из 11 таблиц: Ticket, Session, Film, Genre, Place, Hall, TypePlace, Users, Favorite, Auditt, Token. Ниже приведены их описания, структура и связи. Данные в таблицы были внесены во время тестирования.

Таблица Ticket хранит список проданных билетов на данный момент на доступные сеансы, состоит из столбцов (таблица 2.1).

Таблица 2.1 – Столбцы таблицы Ticket

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор билета, первичный ключ | число |
| IDSession | Идентификатор сеанса, внешний ключ для связи с таблицей «Session» | число |
| IDPlace | Идентификатор места, внешний ключ для связи с таблицей «Place» | число |
| IDUsers | Идентификатор данных пользователя, внешний ключ для связи с таблицей «Users» | число |
| QR | QR-code для прохода на сеанс | строка |

Таблица Users данные авторизации пользователей (таблица 2.2).

Таблица 2.2 – Столбцы таблицы Users

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор пользователя, первичный ключ | число |
| Password | Пароль пользователя | строка |
| Name | Логин пользователя | строка |
| Email | Электронная почта пользователя | строка |

Таблица Genre содержит список всех жанров (таблица 2.3).

Таблица 2.3 – Столбцы таблицы Genre

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор жанра, первичный ключ | число |
| Name | Название жанра | строка |

Таблица Film содержит список всех фильмов, внесенных в базу (таблица 2.4).

Таблица 2.4 – Столбцы таблицы Film

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор фильма, первичный ключ | число |
| Name | Название фильма | строка |
| StartRelease | Дата начала проката фильма | дата |
| EndRelease | Дата окончания проката фильма | дата |
| Description | Описание фильма | строка |
| AgeLimit | Возрастное ограничение | число |
| IDGenre | Идентификатор жанра, внешний ключ для связи с таблицей «Genre» | число |

Продолжение таблицы 2.4

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| IMG | Афиша фильма | число |
| Year | Год выпуска фильма | число |
| Duration | Продолжительность фильма | время |

Таблица Hall содержит информацию о залах кинотеатра (таблица 2.5).

Таблица 2.5 – Столбцы таблицы Hall

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор зала, первичный ключ | число |
| Name | Название зала | строка |
| Capacity | Общее число мест в зале | число |

Таблица Session содержит сеансы фильмов, которые сейчас в прокате в кинотеатре (таблица 2.6).

Таблица 2.6 – Столбцы таблицы Session

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор пользователя, первичный ключ | число |
| DateSession | Дата сеанса | дата |
| TimeSession | Время сеанса | время |
| IDHall | Идентификатор зала, внешний ключ для связи с таблицей «Hall» | число |
| IDFilm | Идентификатор фильма, внешний ключ для связи с таблицей «Film» | число |

Таблица Place содержит информацию о местах в залах (таблица 2.7).

Таблица 2.7 – Столбцы таблицы Place

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор места, первичный ключ | число |
| Place | Номер места | число |
| IDHall | Идентификатор зала, внешний ключ для связи с таблицей «Hall» | число |

Таблица TypePlace включает в себя список всех типов мест (таблица 2.8).

Таблица 2.8 – Столбцы таблицы TypePlace

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор типа места | число |
| Cost\_vip | Стоимость вип места | число |
| Cost\_normal | Стоимость обычного места | число |

Таблица Favorite включает в себя список избранных фильмов (таблица 2.9).

Таблица 2.9 – Столбцы таблицы Favorite

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор любимого фильма | число |
| IDUsers | Идентификатор пользователя | число |
| IDFilm | Идентификатор фильма | число |

Таблица AUDITT включает в себя данные действий всех пользователей приложения (таблица 2.10).

Таблица 2.10 – Столбцы таблицы AUDITT

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| OperationType | Тип действия | число |
| FilmName | Название фильма | строка |
| Date | Дата изменения | дата |
| Datenow | Текущая дата | дата |
| Place | Место | строка |
| Users | Пользователь, выполнивший действие | строка |

Таблица Token включает в себя список токенов (таблица 2.11).

Таблица 2.11 – Столбцы таблицы Token

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор токена | число |
| IDUsers | Идентификатор пользователя | число |
| Token | Токен | строка |

В данном пункте были рассмотрены структуры всех таблиц базы данных.

2.7 Выводы по разделу

В этом разделе подробно рассмотрели процесс проектирования базы данных, клиентской и серверной частей приложения. Так же мы расписали основные сущности базы данных(таблицы), расписали их свойства(поля). В данной базе данных содержится одиннадцать таблиц, которые принадлежат третьей нормальной форме, что позволяет быстро и качественно хранить, обрабатывать и получать информации, что впоследствии способствует быстрой и бесперебойной работе приложения. Были рассмотрены технологии, которые использовались для разработки приложения, их основные достоинства и недостатки. Был выбран строго типизированный язык NestJS, что позволит обеспечить безопасность и надежность моего приложения, а React позволит создать качественный и простой в восприятии пользовательский интерфейс, при минимальных затратах в ресурсах.