Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение высшего образования

«Хабаровский государственный университет экономики и права»

Факультет управления и технологий

Кафедра информационных систем и технологий

КУРСОВАЯ РАБОТА

по дисциплине «Разработка приложений баз данных»

на тему «Разработка клиентской части онлайн-приложения для покупки билетов в кино»

Студент группы     ПИ(б)-71       \_\_\_\_\_\_     \_\_\_\_\_\_\_\_\_\_\_\_       Драган Е.В.

                                 номер группы                      дата                              подпись

Научный руководитель  \_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_   Е. А. Николаев

                                                                                    уч. степень,                                 уч. звание

Дата регистрации «\_\_\_\_»\_\_\_\_\_\_\_ 20\_\_\_\_г. № \_\_

Решение руководителя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                         (к защите, на доработку)

Подпись \_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_ 20 \_\_\_г.

Дата регистрации после доработки:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 20\_ \_\_г.   № \_\_\_

Решение руководителя

после доработки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                         (к защите, на доработку)

Оценка научного руководителя \_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                                        (удовлетворительно и т.д.)

Подпись \_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_\_» \_\_\_\_\_\_\_\_ 20 \_\_г.

Хабаровск 2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc57557213)

[1 Объектно-ориентированный подход к проектированию баз данных 5](#_Toc57557214)

[1.1 Основы объектно-ориентированной методологии 5](#_Toc57557215)

[1.2 Принципы объектного подхода 7](#_Toc57557216)

[1.3 Объектно-ориентированные базы данных 8](#_Toc57557217)

[1.4 Достоинства модели объектно-ориентированных баз данных 11](#_Toc57557218)

[1.5 Недостатки модели объектно-ориентированных баз данных 13](#_Toc57557225)

[2 Разработка клиентской части интернет-сервиса для покупки билетов в кино 17](#_Toc57557237)

[2.1 Общие сведения 17](#_Toc57557238)

[2.2 Выбор средств реализации 18](#_Toc57557239)

[2.3 Разработка клиентской части 20](#_Toc57557240)

[ЗАКЛЮЧЕНИЕ 26](#_Toc57557241)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc57557242)

# ВВЕДЕНИЕ

Максимум комфорта и минимум времени – своеобразный стиль существования современного человека.

Онлайн сервисы – это сайты, которые предоставляют всевозможные услуги, что значительно облегчает работу и позволяет существенно сэкономить время. При помощи таких сайтов в Интернете можно делать что угодно: проводить денежные операции, общаться, искать, хранить, редактировать, пересылать и публиковать информацию, и многое другое.

Невзирая на все разнообразие сайтов и услуг, которые они оказывают, онлайн сервисы можно разделить на следующие основные группы, в зависимости от их тематики и назначения:

* Информационно-поисковые.
* Социальные.
* Сервисы покупок.
* Банковские сервисы.

Задачей данной курсовой работы является разработка клиентской части онлайн-приложения для покупки билетов в кино. Приложение позволит не только осуществлять покупку билетов онлайн, но также информирует пользователей о близжайших сеансах на неделю и о наличии доступных билетов.

Целью данной работы является приобретение навыков разработки приложения для работы с базой данных, с использованием объектно-ориентированного и визуального программирования.

Для выполнения курсовой работы были поставлены следующие задачи:

* провести анализ предметной области;
* разработать приложение для работы с базой данных.

# 1 Объектно-ориентированный подход к проектированию баз данных

* 1. Основы объектно-ориентированной методологии

Объектно-ориентированная технология развивается в различных областях вычислительной техники как средство решения проблем связанных со сложностью создаваемых систем. Объектный подход применяется не только в программировании, но также в проектировании интерфейса пользователя, баз данных, баз знаний и даже компьютерной архитектуры. Смысл такого широкого подхода состоит в том, что он позволяет применить объектную ориентацию для решения всего круга проблем, связанных со сложными системами. В основе объектно-ориентированного проектирования лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов, рассматривая каждый объект как экземпляр определенного класса, причем классы образуют иерархию.

Составными частями объектно-ориентированной методологии (ООМ) являются:

1. Объектно-ориентированный анализ.

Объектно-ориентированный анализ (ООА) направлен на создание моделей, более близких к реальности, с использованием объектноориентированного подхода. Это методология, при которой требования формируются на основе понятий классов и объектов, составляющих словарь предметной области. На результатах ООА формируются модели, на которых основывается объектно-ориентированное проектирование. Объектно-ориентированное проектирование в свою очередь создает основу для окончательной реализации системы с использованием методологии объектно-ориентированного программирования.

1. Объектно-ориентированное проектирование.

Методы программирования, прежде всего, подразумевают правильное и эффективное использование механизмов языков программирования. Методы проектирования напротив, основное внимание направляют на правильное и эффективное структурирование сложных систем.

Объектно-ориентированное проектирование – это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления как логической и физической, так статической и динамической моделей проектируемой системы.

Именно поддержка объектно-ориентированной декомпозиции отличает объектно-ориентированное проектирование от структурного проектирования.

1. Объектно-ориентированное программирование.

##### Обьектно-ориентированное программирование – это методология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса, а классы образуют иерархию на принципах наследования.

##### Главными достоинствами ООМ по сравнению со структурными методами являются:

##### возможность преодолеть ограничения, связанные со сложностью разрабатываемых систем;

##### использование на стадии анализа моделей близких к реальности;

##### обеспечение возможности повторного использования разработанного программного обеспечения, позволяющего существенно сократить сроки и снизить затраты на разработку каждой последующей системы;

##### поддержка итеративного, а не лавинообразного, как в структурном подходе, процесса проектирования;

##### создание более открытых систем;

##### полное использование описательных возможностей объектно-ориентированных языков программирования.

1.2 Принципы объектного подхода

Объектная модель, которая является концептуальной базой объектно-ориентированной методологии, имеет четыре главных элемента:

* абстрагирование;
* ограничение доступа или инкапсуляция;
* модульность;
* иерархия.

Без любого из этих элементов модель не будет объектно-ориентированной. Кроме главных имеется три дополнительных элемента:

* типизация;
* параллелизм;
* сохраняемость или устойчивость.

Эти элементы полезны в объектной модели, но не обязательны. Абстрагирование - это выделение таких существенных характеристик объекта, которые отличают его от всех других видов объектов и таким образом чётко определяются особенности данного объекта с точки зрения дальнейшего его рассмотрения.

Инкапсуляция – это процесс разделения элементов объекта, определяющих его устройство и поведение. Инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.

Иерархия – упорядоченная система абстракций. Основными видами иерархических структур, применительно к сложным системам, является структура классов (иерархия "is-a") и структура объектов (иерархия "part of"). Принцип наследования позволяет упростить выражения абстракции, делая проект менее громоздким и более выразительным.

Наследование – это такая иерархичность абстракций, в которой подклассы наследуют строение от одного или нескольких суперклассов. В подклассе, кроме того, могут быть определены дополнительные атрибуты и методы. Суперклассы отражают наиболее общие, а подклассы более специализированные абстракции.

Дополнительные элементы:

Типизация – ограничение предъявляемых классу объектов, препятствующих взаимозамене различных классов и в большинстве случаев сильно сужающих возможность такой замены. Концепция типизации строится на понятии абстрактных типов данных. Тип - точное определение свойств строения или поведения, которое присуще некоторой совокупности объекта. Часто термины «тип» и «класс» считают эквивалентными. Более точно сказать, что класс реализует тип.

Полиморфизм возникает на стыке принципов наследования и динамических связей. Это свойство является самым существенным в объектно-ориентированном программировании. Полиморфизм отличает объектно-ориентированное проектирование от более традиционных методов с использованием абстрактных типов данных.

Устойчивость - это свойство объекта существовать во времени и/или пространстве, вне зависимости от процессов, породивших данный объект.

## 1.3 Объектно-ориентированные базы данных

К настоящему моменту терминология еще не устоялась, существует много разных определений и трактовок. Представляется, что объектно-ориентированная база данных (ООБД) – база данных, основанная на принципах объектно-ориентированной технологии. К основным описательным моментам, связанным с ООБД, в относят:

1. объекты (в ООБД любая сущность – объект и обрабатывается как объект);
2. классы (понятие "тип данных" реляционной модели заменяется понятиями "класс" и "подкласс");
3. наследование (классы образуют иерархию наследования, заимствуя свойства друг друга);
4. атрибуты (характеристики объекта моделируются его атрибутами);
5. сообщения и методы (каждый класс имеет определенную совокупность методов, классы взаимодействуют друг с другом посредством механизма сообщений);
6. инкапсуляция (внутренняя структура объектов скрыта);
7. идентификаторы объектов – дескрипторы.

  
Рисунок 1 -  Схема представления объекта

Поскольку каждый объект данного класса имеет один и тот же набор методов, методы сохраняются только один раз – как методы класса (данные каждого экземпляра объекта хранятся отдельно).



Рисунок 2 – Схема представления класса объектов

ODMG (Object Data Management Group) – консорциум поставщиков ООБД и других заинтересованных организаций, созданный в 1991 г. Его задачей является разработка стандарта на хранение объектов в базах даннных.

Стандарт на хранение объектов ODMG разработан на основе трех существующих стандартов: управление базами данных (SQL), объекты (стандарты OMG - Object Management Group) и стандарты на объектно-ориентированные языки программирования (C++, Smalltalk, Java).

ODMG добавляет возможности взаимодействия с базами данных в объектно-ориентированные языки программирования: определяются средства долговременного хранения объектов и расширяется семантика языка, вносятся операторы управления данными. Стандарт состоит из нескольких частей:

1. Объектная модель - унифицированная основа всего стандарта. Она расширяет объектную модель консорциума OMG за счет введения таких свойств как связи и транзакции для обеспечения функциональности, требуемой при взаимодействии с базами данных. Ключевые концепции объекной модели ODMG:
   * наделение объектов такими свойствами как атрибуты и связи
   * методы объектов (поведение)
   * множественное наследование
   * идентификаторы объектов (ключи)
   * определение таких совокупностей объектов как списки, наборы, массивы и т.д.
   * блокировка объектов и изоляция доступа
   * операции над базой данных
2. Язык описания объектов (ODL) - средство определения схемы базы данных (по аналогии с DDL в реляционных СУБД). ODL является расширением IDL (Interface Definition Language - язык описания интерфейсов) модели OMG и предоставляет средства для определения объектных типов, их атрибутов, связей и методов. ODL создает слой абстрактных описаний так, что схема базы данных становится независима как от языка программирования, так и от СУБД. ODL рассматривает только описание объектных типов данных, не вдаваясь в детали реализации их методов. Это позволяет переносить схему БД между различными ODMG-совместимыми СУБД и языками программирования, а также транслировать ее в другие DDL.
3. Язык объектных запросов (OQL) – SQL-подобный декларативный язык, который предоставляет эффективные средства для извлечения объектов из базы данных, включая высокоуровневые примитивы для наборов объектов и объектных структур.
4. Связывание с ОО-языками. Стандарт связывания с C++, Smalltalk и Java определяет Object Manipulation Language (OML) - язык манипулирования объектами, который расширяет базовые ОО-языки средствами манипулирования и хранения объектов. Также включаются OQL, средства навигации и поддержка транзакций. Каждый ОО-язык имеет свой собственный OML, поэтому разработчик остается в одной языковой среде, ему нет необходимости разделять средства программмирования и доступа к данным.

## 1.4 Достоинства модели объектно-ориентированных баз данных

Объектно-ориентированные базы данных позволяют представлять сложные объекты более непосредственным образом, нежели реляционные системы. Остановимся на некоторых имеющихся достижениях в области ООБД.

Системы ООБД позволяют пользователям:

* определять абстракции;
* облегчают проектирование некоторых связей;
* устраняют потребность в определяемых пользователями ключах;
* поддерживают новый набор предикатов сравнения;
* в некоторых случаях устраняют потребность в соединениях;
* в некоторых ситуациях обеспечивают более высокую производительность, нежели системы, основанные на реляционной модели;
* обеспечивают поддержку версий и длительных транзакций.

### 1) Пользовательские абстракции

Объектно-ориентированные базы данных предоставляют возможность определять новые абстракции и управлять реализацией таких абстракций. Эти новые абстракции могут соответствовать структурам данных, требуемым в сложных задачах, новым абстрактным типам данных. Иначе говоря, современные пакеты ООБД дают пользователю возможность создания нового класса с атрибутами и методами, иметь классы, наследующие атрибуты и методы от классов-предков, создавать экземпляры класса, каждый из которых обладает уникальным объектным идентификатором, извлекать эти экземпляры по одному или группами, а также загружать и выполнять методы.

2) Инверсные связи

В объектно-ориентированных базах данных поддерживается средство инверсных связей для выражения взаимных ссылок между двумя объектами (бинарная связь). Такая система обеспечивает ссылочную целостность путем установления соответствующей обратной ссылки сразу же после создания прямой ссылки. Существует даже возможность автоматического распространения удалений через эти ссылки. Примером пакета ООБД, обеспечивающего автоматическое поддержание инверсных связей, является ObjectStore.

### 3) Первичные ключи

В связи с обязательным условием уникальной идентификации объектов ООБД отпадает необходимость первичных ключей, эта задача решается автоматически.

### 4) Предикаты сравнения

В РБД сравнение всегда базируется лишь на значениях. В этой модели два кортежа являются одной сущностью, если все их ключевые атрибуты имеют одинаковые значения. Однако в модели ООБД были разработаны и определены иные типы сравнения.

### 5) Производительность

ООБД обладают несколькими особенностями, обеспечивающими их выигрыш в производительности.

### 6) Версии и длительные транзакции

В РБД не поддерживаются ни работа с версиями, ни длительные транзакции. Такая поддержка имеется в некоторых ООБД, хотя и с ограниченными возможностями.

### 7)Объектная алгебра

Объектная алгебра не столь подробно разработана и не является столь же зрелой, как реляционная алгебра. Но как бы то ни было, такая алгебра существует, и в ней определяются пять фундаментальных операций, сохраняющих объекты: union, difference, select, generate и map.

## 1.5 Недостатки модели объектно-ориентированных баз данных

В объектно-ориентированных базах данных отсутствуют базовые средства, к которым пользователи систем баз данных привыкли и поэтому ожидают видеть. Среди прочего, можно отметить:

* минимальную оптимизацию запросов;
* отсутствие стандартной алгебры запросов;
* отсутствие средств обеспечения запросов;
* отсутствие поддержки представлений;
* проблемы с безопасностью;
* отсутствие поддержки динамических изменений определений классов;
* ограниченная поддержка ограничений целостности;
* ограниченные возможности настройки производительности;
* недостаточная поддержка сложных объектов;
* ограниченная интеграция с существующими объектно-ориентированными системами программирования;
* ограниченный выигрыш в производительности.

### Оптимизация запросов

Одной из самых значительных проблем в ООБД является оптимизация декларативных запросов. Оптимизацию запросов к ООБД затрудняет дополнительная сложность самой объектно-ориентированой модели данных.

### Алгебра запросов

Еще один серьезный недостаток ООБД состоит в отсутствии стандартов алгебры запросов. Это обстоятельство тоже затрудняет оптимизацию запросов.

### Средства обеспечения запросов

В большинстве ООБД не хватает средств обеспечения запросов.

### Представления

В ООБД не поддерживаются представления. Хотя по этому поводу было выдвинуто несколько предложений, не удалось придти к единому мнению относительно того, как механизм представлений должен функционировать в ООБД.

### 5) Безопасность

В некоторых системах ООБД пользователи должны явным образом устанавливать и снимать блокировки. Системы РБД автоматически устанавливают и снимают блокировки при обработке пользовательских запросов и операторов обновления.

### 6) Динамическое изменение определений классов

В дополнение к тому, что для ООБД до сих пор не разработана единая стандартная модель данных, в большинстве ООБД не допускаются динамические изменения схемы баз данных, таких как добавление к классу нового атрибута или метода, добавление к классу нового суперкласса, удаление суперкласса класса, добавление нового класса и удаление класса. РБД позволяют пользователю динамически изменять схему базы данных с помощью команды ALTER.

### 7) Ограничения целостности

Отсутствуют механизмы объявления ключевых свойств атрибутов (например, атрибут класса не может быть объявлен первичным ключом класса), или ограничений уникальности, явных ограничений целостности, а также пред- и постусловий методов.

### 8) Настройка производительности

В большинстве ООБД имеются лишь ограниченные средства параметризованной настройки производительности.

### 9) Поддержка сложных объектов

Полная функциональность сложных объектов все еще не поддерживается. Можно осуществлять навигацию по ссылкам и кодировать операции с применением этих ссылок, но отсутствуют предопределенные родовые операции, в которых используются различные виды семантики ссылок. Считается, что все ссылки указывают на независимые объекты, а семантика особых связей внутри сложных объектов скрыта в операциях, предоставляемых пользователями.

### Интеграция с объектно-ориентированными системами программирования

Трудно переписывать объектно-ориентированные программы для управления стабильными данными. Здесь возникает ряд проблем: конфликты по именам; необходимость переделывать иерархии классов; склонность ООБД к перегрузке системных операций.

### 11) Производительность

Если бы всем приложениям баз данных требовались только поиск объектов базы данных через идентификаторы объектов и быстрая работа с объектами в основной памяти с использованием указателей, то ООБД действительно превосходили бы РБД по производительности на два-три порядка. Однако для большинства приложений, которым требуется доступ к объектам через идентификаторы, нужны также и возможности доступа к базе данных и ее обновления, обеспечиваемые в РБД. В число этих возможностей входят массовая загрузка базы данных; создание, обновление и удаление индивидуальных объектов (по одному); извлечение из класса одного или более объектов, удовлетворяющих определенным условиям поиска; соединение нескольких классов; фиксация транзакций и т.д. При выполнении таких приложений ООБД не имеют никаких преимуществ в производительности по сравнению с РБД.

В числе средств, пока что не поддерживаемых в ООБД, также можно назвать триггеры, средства управления метаданными, а также ограничения целостности, такие как UNIQUE и NULL.

Подведем некоторые итоги:

В объектно-ориентированных базах данных, в отличие от реляционных, хранятся не записи, а объекты. ОО-подход представляет более совершенные средства для отображения реального мира, чем реляционная модель:

* естественное представление данных.
* имеется возможность определения новых типов данных и операций с ними.

В то же время, ОО-модели присущ и ряд недостатков:

* осутствуют мощные непроцедурные средства извлечения объектов из базы. Все запросы приходится писать на процедурных языках, проблема их оптимизации возлагается на программиста.
* вместо чисто декларативных ограничений целостности или полудекларативных триггеров для обеспечения внутренней целостности приходится писать процедурный код.

Очевидно, что оба эти недостатка связаны с отсутствием развитых средств манипулирования данными. Эта задача решается двумя способами - расширение ОО-языков в сторону управления данными (стандарт ODMG), либо добавление объектных свойств в реляционные СУБД (SQL-3, а также так называемые объектно-реляционных СУБД).

1. Разработка клиентской части интернет-сервиса для покупки билетов в кино
   1. Общие сведения

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер».

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Веб-интерфейс – это совокупность средств, при помощи которых пользователь взаимодействует с  веб-сайтом или веб-приложением через браузер. Веб-интерфейсы получили широкое распространение в связи с ростом популярности всемирной паутины и соответственно – повсеместного распространения веб-браузеров.

Одним из основных требований к веб-интерфейсам является их одинаковый внешний вид и одинаковая функциональность при работе в различных браузерах.

Варианты реализации:

Классическим и наиболее популярным методом создания веб-интерфейсов является использование HTML с применением CSS и JavaScript. Однако различная реализация HTML, CSS, DOM и других спецификаций в браузерах вызывает проблемы при разработке веб-приложений и их последующей поддержке. Кроме того, возможность пользователя настраивать многие параметры браузера (например, размер шрифта, цвета, отключение поддержки сценариев) может препятствовать корректной работе интерфейса.

Другой (менее универсальный) подход заключается в использовании Adobe Flash, Silverlight или Java-апплетов для полной или частичной реализации пользовательского интерфейса. Поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов), Flash- или Java-приложения могут выполняться с легкостью. Так как они предоставляют программисту больший контроль над интерфейсом, они способны обходить многие несовместимости в конфигурациях браузеров, хотя несовместимость между Java или Flash реализациями на стороне клиента может приводить к различным осложнениям.

В настоящее время набирает популярность новый подход к разработке интерфейсной части веб-приложений, называемый Ajax. При использовании Ajax интерфейсы не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Было решено разрабатывать онлайн-сервис для покупки билетов в кино. Целью данного web-сервиса является онлайн покупка и оплата билетов в кино и просмотр близжайших сеансов.

Данное web-приложение рассчитано на массового пользователя. В частности, такой сервис актуален для всех людей, посещающих кинотеатры в городе Хабаровске.

* 1. Выбор средств реализации

Для реализации поставленной задачи были выбраны следующие технологии:

* 1. Microsoft visual code

Visual Studio Code – редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

* 1. React.js

React - это декларативная, эффективная и гибкая библиотека JavaScript для создания пользовательских интерфейсов (UI). Она позволяет вам создавать сложные UI из небольших и изолированных частей кода, называемых «компонентами».

* 1. Bootstrap

Bootstrap – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Bootstrap использует современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров.

* 1. Moment.js

Moment.js это отличная библиотека для работы с датами в JavaScript.

* 1. Axios

Axios – это широко известная JavaScript-библиотека. Она представляет собой HTTP-клиент, основанный на промисах и предназначенный для браузеров и для Node.js.

* 1. Системы управления версиями, Github

GitHub – крупнейшийвеб-сервис для хостинга IT-проектов и их совместной разработки.

Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

* 1. Разработка клиентской части

Для того, чтобы начать разработку веб-приложения, необходимо создать React-приложение и установить все необходимые технологии с помощью npm (менеджер пакетов, входящий в состав Node.js.)

Для создания React-приложения прежде всего понадобится node.js и терминал.

Установим create-react-app в терминале (рисунок 3):



Рисунок 3 – Установка create-react-app

Создание приложения (рисунок 4).

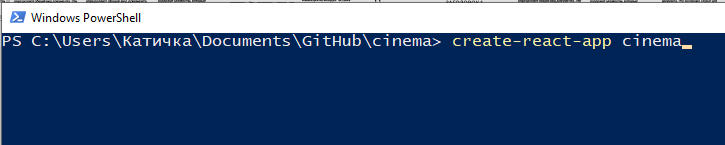


Рисунок 4 – Создание приложения

Запуск приложения (рисунок 5).

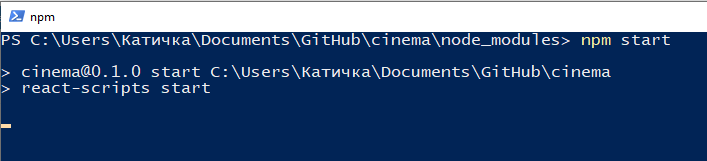


Рисунок 5 – Запуск приложения

Новое приложение автоматически откроется в браузере.

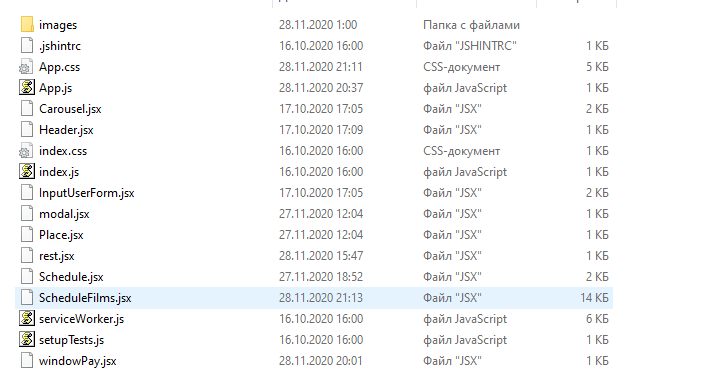


Рисунок 6 – Структура проекта

На рисунке 6 показана структура проекта.

В файле rest.jsx находятся запросы к сервесу. Компоненты приложения это файлы: App.jsx, Header.jsx, Schedule.jsx, ScheduleFilm.jsx, WindowsPay.jsx и тд.

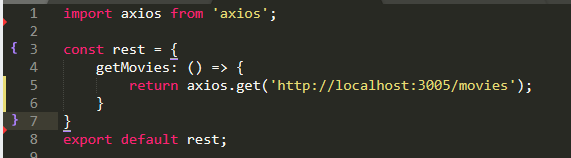


Рисунок 7 – Пример запроса к серверу

Далее рассмотрим функционал уже созданного приложения для покупки билетов в кино.



Рисунок 8 – Шапка сайта

На рисунке 6 представлена шапка сайта, которая включает в себя логотип, номер телефона и карусель, реализованную с помощью Bootstrap.

Шапка сайта состоит из двух компонентов:

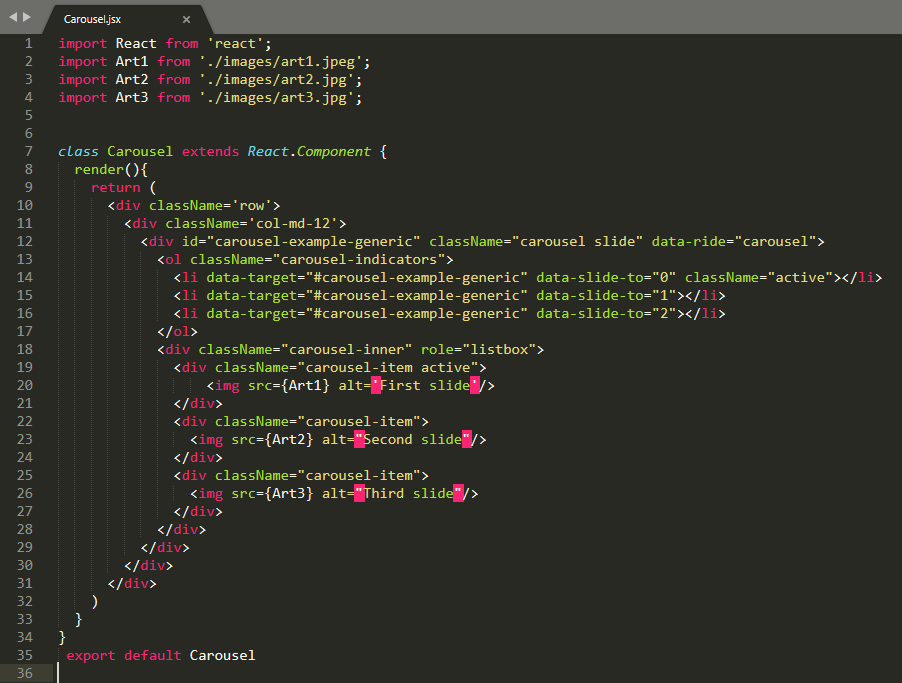


Рисунок 9 – Компонент Carousel.jsx

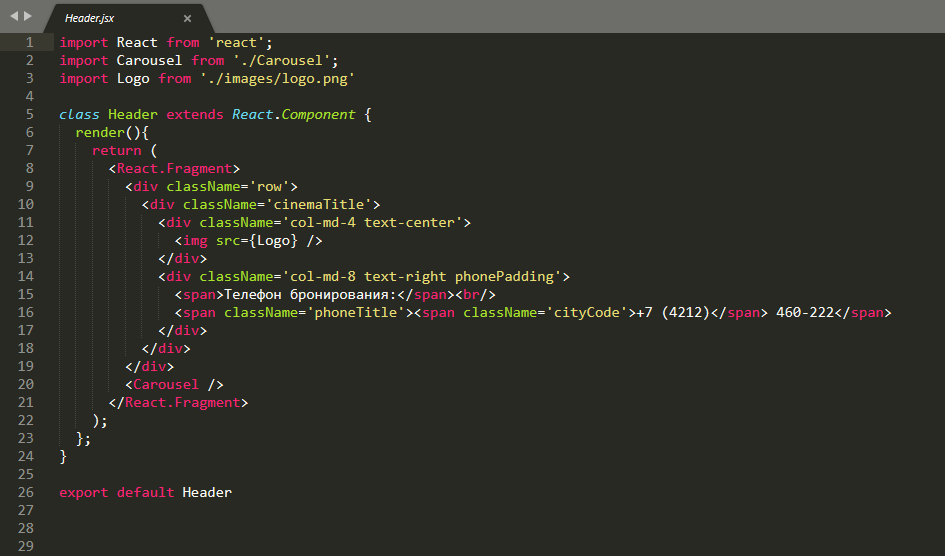


Рисунок 10 – Компонент Header.jsx

Следующий компонент на странице это меню с датами. Которое динамически обновляется отталкиваясь от сегодняшней даты, которую с помощью библиотеки Moment.js можно успешно получить.



Рисунок 11 – Меню с датами фильмов

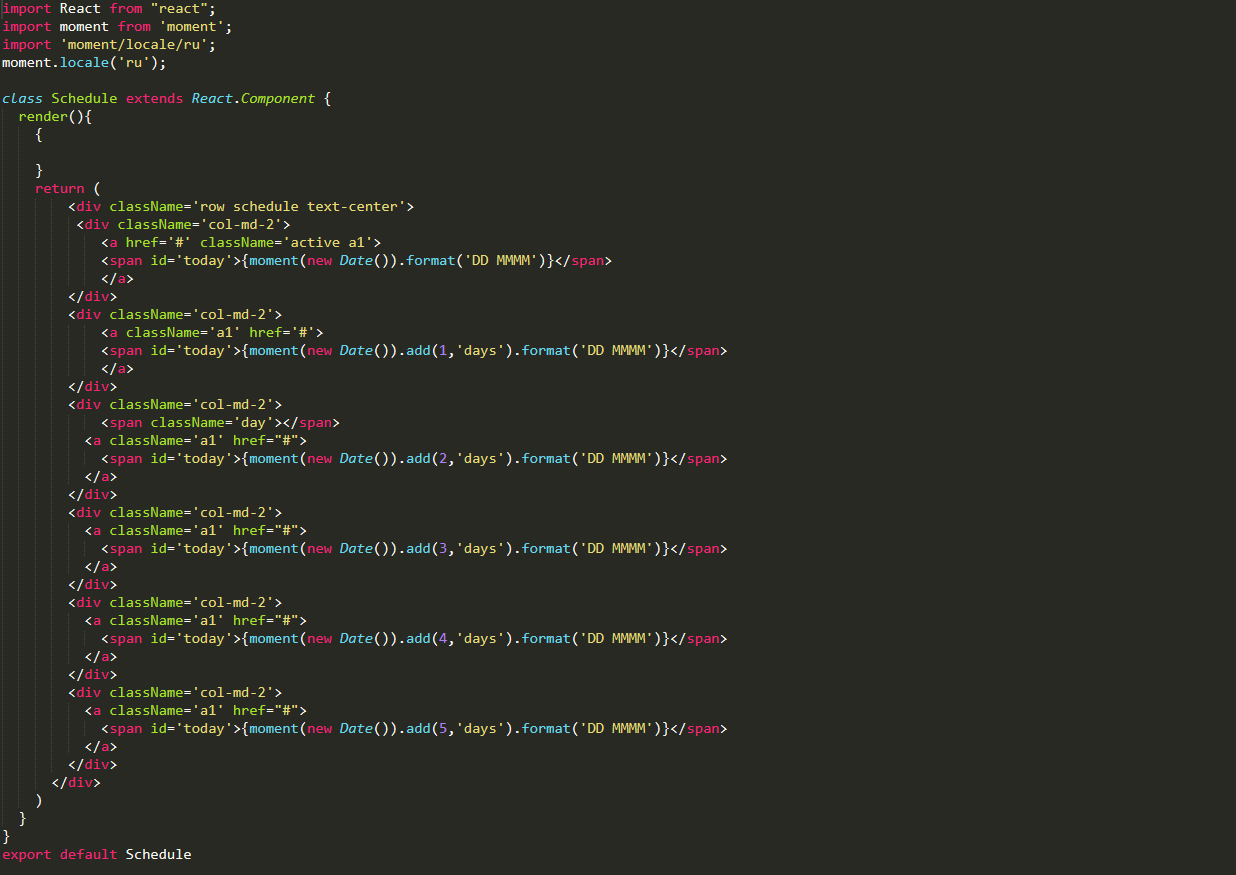


Рисунок 12 – Листинг компонента Schedule

Данная библиотека предоставляет много возможностей для работы с данными. Например, можно вывести дату в любом удобном формате.

Пользователь может выбрать нужную ему дату и посмотреть расписание сеансов. Все данные о фильмах и сеансах хранятся в БД и с помощью запросов к серверу мы получаем данные и динамически выводим их для пользователя (рисунок 10).

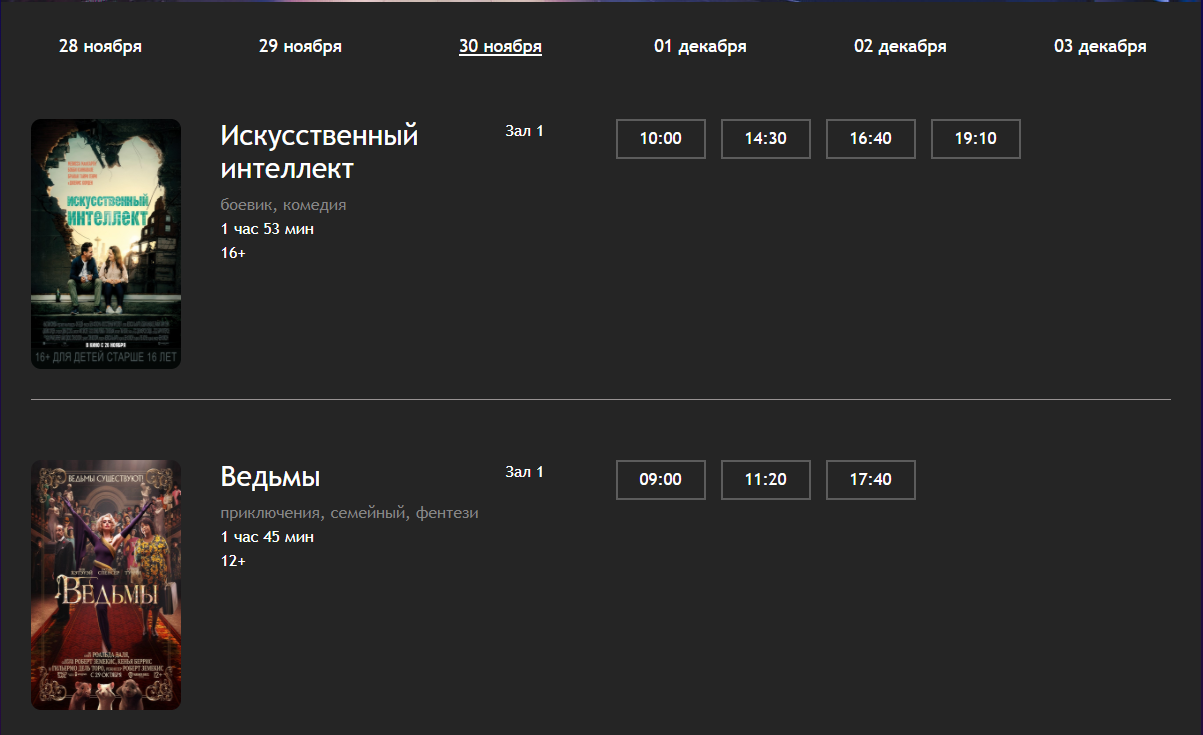


Рисунок 13 – Расписание сеансов

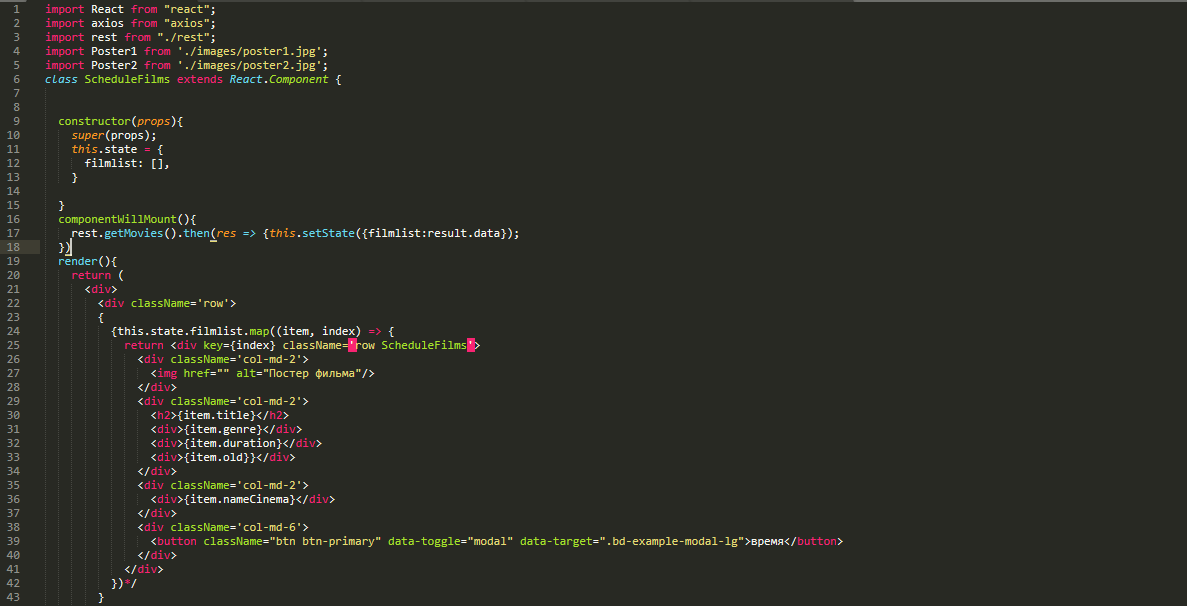


Рисунок 14 – Листинг компонента раписание сеансов

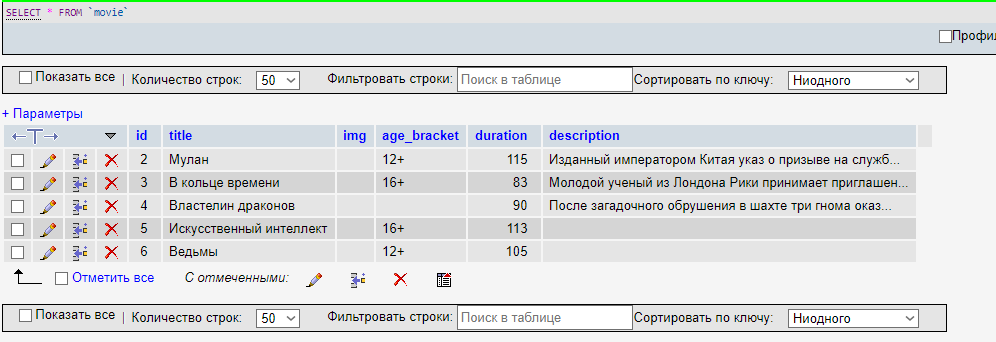


Рисунок 15 – Таблица с фильмами

Здесь (рисунок 10) само время киносеанса является кнопкой, которая открывает модульное окно.

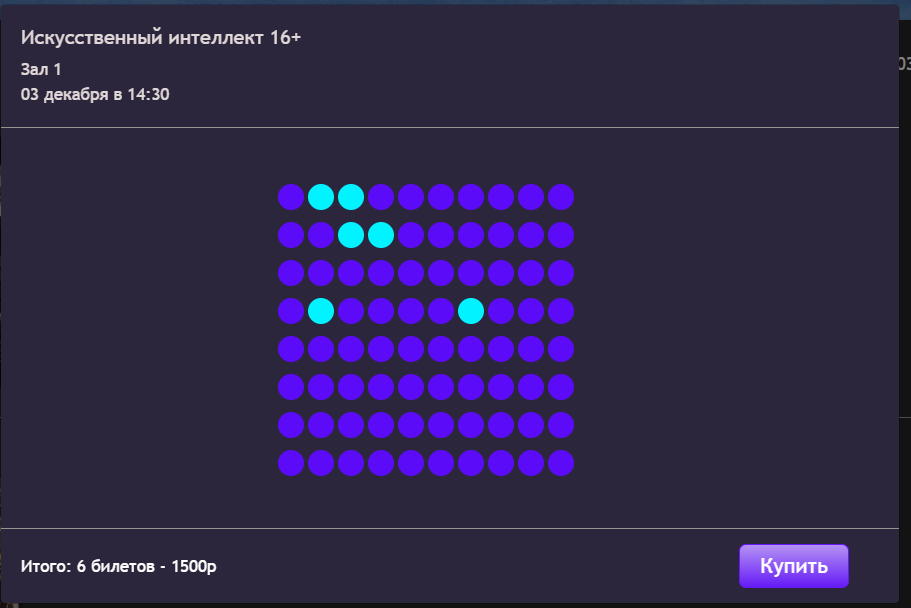


Рисунок 16 – Модульное окно

На модульном окне отображается название фильма, зал и дата сеанса. Самое главное, что мы видим это интерактивная схема зала. Нужно выбрать места нажав на них, выбранное место поменяет цвет на голубой. В левом нижнем углу можно увидеть итоговую сумму, а в правом кнопку «купить», которая перенаправит на следующее окно «оформление покупки».

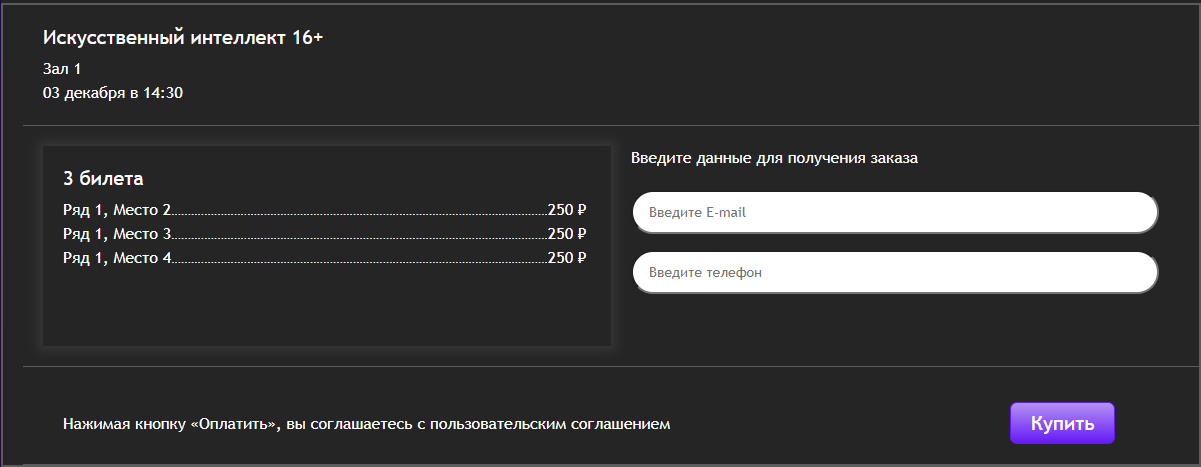


Рисунок 17 – Окно оформления заказа

В данном окне (рисунок 12) пользователь может увидеть на какие места он выбрал билеты, затем ему нужно ввести свой номер телефона и электронный адрес. По нажатию на кнопку «купить» должно открываться окно для ввода данных своей кредитной карты, но в рамках данной курсовой работы это не было реализовано. На данном этапе при нажатии кнопки «Купить» производится занесение покупки в базу данных.

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы была разработана клиентская часть приложения для кинотеатра. Данное приложение не является законченной, полнофункциональной программой. В качестве дальнейшего развития возможна и необходима доработка, а именно подключение платежной системы.В перспективах развития этой темы можно реализовать мобильную версию приложения.

Были приобретены навыки разработки приложений с использованием объектно-ориентированного и визуального программирования.

В результате разработки приложения для кинотеатра были решены следующие задачи:

* был проведен анализ предметной области;
* были выбраны и использованы инструменты по разработке приложений;
* была разработан клиентская часть приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Бойко Т. С. Рожков Ю. В. Научные работы: учебно-методическое пособие по написанию и оформлению научных работ для студентов, магистрантов, аспирантов всех форм обучения и специальностей. – Хабаровск: РИЦ ХГАЭП, 2020. – 76 с.

2 Мардан А. React быстро. Веб-приложения на React, JSX, Redux и GraphQL– Питер, 2019. - 228c.

3 Мэтью, Дэвид HTML5. Разработка веб-приложений / Дэвид Мэтью. - М.: 4 Рид Групп, 2018. - 320 c

4 Роббинс, Дженнифер HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженнифер Роббинс. - М.: Эксмо, 2018. - 310 c.

5 URL: https://learn-reactjs.ru/tutorial (дата обращения 20.10.2020)

6 URL: https://prognote.ru/ajax-in-axios-js/ (дата обращения 03.11.2020)

7 URL: https://momentjs.com (дата обращения 05.11.2020)

8 URL: https://javascript.ru (дата оращения 15.10.2020)

9 URL: https://ru.reactjs.org (дата обращения 25.10.2020)

10 Мержевич В.И. HTML и CSS на примерах. – С.Пб.: БХВ-Петербург, 2015.- 127с.