МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Вятский государственный университет»**

**Колледж ВятГУ**

**КУРСОВОЙ ПРОЕКТ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| По междисциплинарному курсу | | | | | МДК 05.01 Проектирование и дизайн информационных | | | | |
| систем | | | | |
| Тема | Разработка курсового проекта на создание информационной системы для | | | | | | | | | |
| кинотеатра | | | | | | | | | |
| Студента (ки) | | | | Осколкова Максима Владимировича | | | | | | |
|  | | | *ФИО (полностью в родительном падеже)* | | | | | | | |
| Курс | | 3 | | | | Форма обучения | очная | |
|  | | *(арабской цифрой)* | | | |  | | *(очная, заочная)* | | |
| Основная профессиональная образовательная программа по специальности | | | | | | | | | | |
| 09.02.07 Информационные системы и программирование | | | | | | | | | | |
| (код и наименование специальности без кавычек) | | | | | | | | | | |

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, инициалы)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(прописью, без сокращений)*

Киров, 2023

**РЕФЕРАТ**

Пояснительная записка к курсовому проекту содержит:31 страницу, 14 рисунков, 8 использованных источников.

Объектом и предметом исследования является информационная система для кинотеатра.

Цель работы – ознакомление с процессом создания технического задания на разработку информационной системы для кинотеатра.

Поставлена задача разработать техническое задание для информационной системы кинотеатра.

В процессе работы были проведены следующие исследования: 1) обзор предметной области, обзор аналогов и сравнительный анализ; 2) написание самого технического задания, состоящего из: назначения разработки, функциональных характеристик, условий эксплуатации и требований к составу и параметрам технических средств; 3) описание решения и концепции; 4) архитектура решения; 5) разработка схем бизнес-процессов с их описанием; 6) разработка схем алгоритмов и кода на естественном языке; 7) проектирование прототипа пользовательского интерфейса с описанием.

Область применения включает в себя предоставление удобного и информативного оффлайн-ресурс для сотрудников. Эффективное приложение сети кинотеатра обеспечивает простоту использования для посетителей и управление для администраторов, способствуя привлечению и удержанию клиентов.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc154377631)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc154377632)

[1.1. Описание предметной области 5](#_Toc154377633)

[1.1.1 Концепция 5](#_Toc154377634)

[1.2. Обзор аналогов 6](#_Toc154377635)

[1.2.1. "1C: Кинотеатр" 6](#_Toc154377636)

[1.2.2. "1C: Театр" 7](#_Toc154377637)

[2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 9](#_Toc154377638)

[3. РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 10](#_Toc154377639)

[3.1. Моделирование ИС 10](#_Toc154377640)

[3.1.1. Разработка IDEF0 10](#_Toc154377641)

[3.1.2. Разработка DFD 12](#_Toc154377642)

[3.1.3. Разработка диаграммы вариантов использования 13](#_Toc154377643)

[3.2. Прототипирование и разработка интерфейса 15](#_Toc154377644)

[3.3. Разработка базы данных 18](#_Toc154377645)

[3.3.1. Разработка логической и физической моделей подсистемы 18](#_Toc154377646)

[4 ЗАКЛЮЧЕНИЕ 20](#_Toc154377647)

[5 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc154377648)

[6 ПРИЛОЖЕНИЕ А 22](#_Toc154377649)

[7 ПРИЛОЖЕНИЕ Б 23](#_Toc154377650)

# **ВВЕДЕНИЕ**

Курсовой проект проходил на базе Колледжа ФГБОУ ВО «Вятский государственный университет» в период с 01.09.2023 г. по 24.12.2023 г.

В документе рассматриваются следующие разделы: Анализ предметной области, Техническое задание, Реализация информационной системы, Заключение, Список использованных источников, Приложения.

В первом разделе содержится описание изучаемой темы, обзор существующих аналогов, их преимущества и недостатки, а также основные проблемы, решаемые в данном проекте. В пункте Техническое задание описывается понятие ТЗ и его роль. Раздел Реализация информационной системы включает в себя этапы создания информационной системы, включая моделирование её работы, разработку прототипа и интерфейса, а также создание базы данных, необходимой для функционирования системы. В следующем пункте подводится вывод о проделанной работе и обобщение полученных результатов. Список использованных источников содержит в себе перечисление всех использованных при написании курсового проекта источников информации. Последний раздел Приложения включает в себя все дополнительные материалы, такие как ТЗ и программный код приложения.

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## Описание предметной области

Предметная область сети кинотеатров охватывает сферу развлечений и кинематографии. ориентирована на обеспечение информирования и удобства посетителей, а также эффективное управление кинотеатрами и продажей билетов.

### Концепция

Система информационной системы кинотеатра должна предоставлять возможность:

* масштабирование окна программы;
* просмотр и редактирование информации о фильмах
* создать расписание с возможностью его редактирования
* просмотр и бронирование свободных мест для конкретного фильма

## Обзор аналогов

### "1C: Кинотеатр"

**Программа предназначена для автоматизации ведения учета в кинотеатрах, театрах, киноклубах и т.п.**

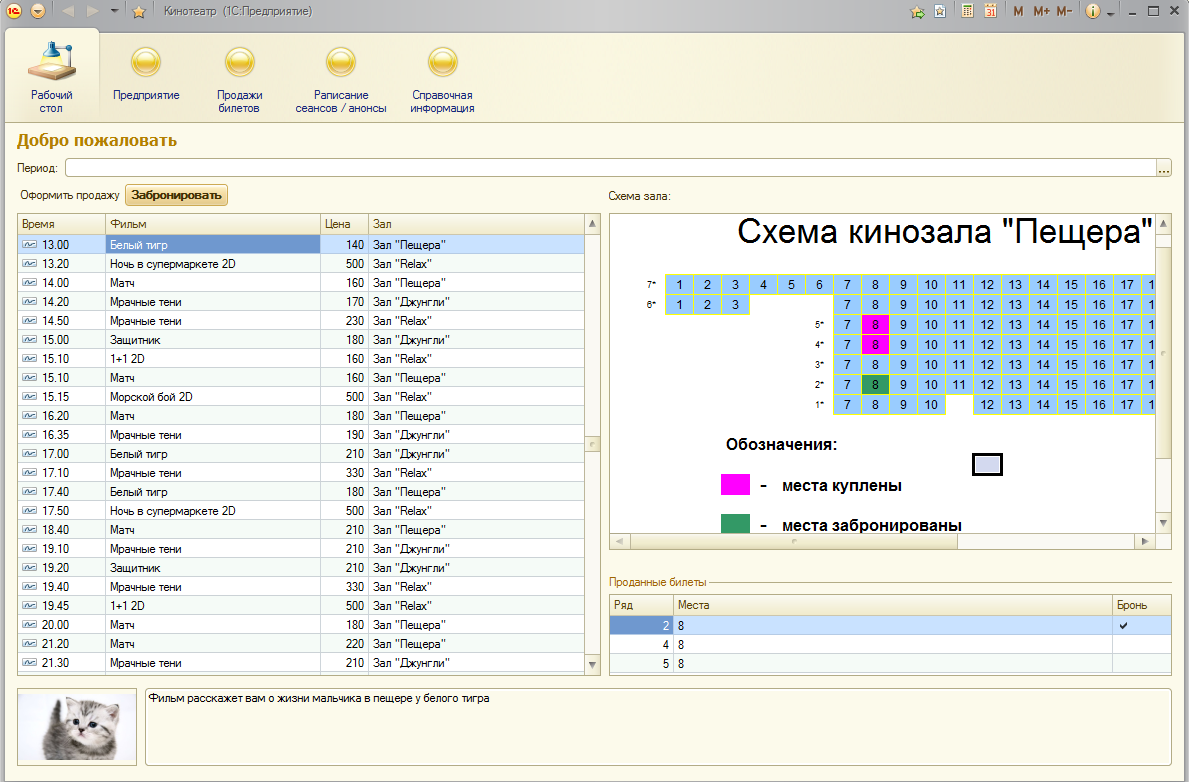


Рисунок 1 – "1C: Кинотеатр"

* **Предприятие.**

В данном разделе сосредоточена информация, касающаяся работы кинотеатра и залов.

* **Продажи билетов.**

Основной раздел, отвечающий за продажу, бронирование, распечатку билетов и чеков.

* **Расписание сеансов**.

Раздел несет в себе справочную информацию о всех фильмах, анонсах и расписаниях.

* **Справочная информация.**

Содержится собранная из всех разделов справочная информация, облегчающая настройку конфигурации

### "1C: Театр"

Программный продукт "1C: Театр" предназначен для комплексной автоматизации управления деятельностью театрального учреждения, включая художественное руководство, руководство постановочной частью и производственными мастерскими, финансово-экономическое управление, управление кадрами.

"1C: Театр" может использоваться в управлении деятельностью театральных, концертных организаций и коллективов всех форм собственности.

Программа предоставляет следующие технологические возможности:

* Работа в режиме управляемого интерфейса;
* Поддержка толстого, тонкого и веб-клиента;
* Использование Библиотеки стандартных подсистем "1С".

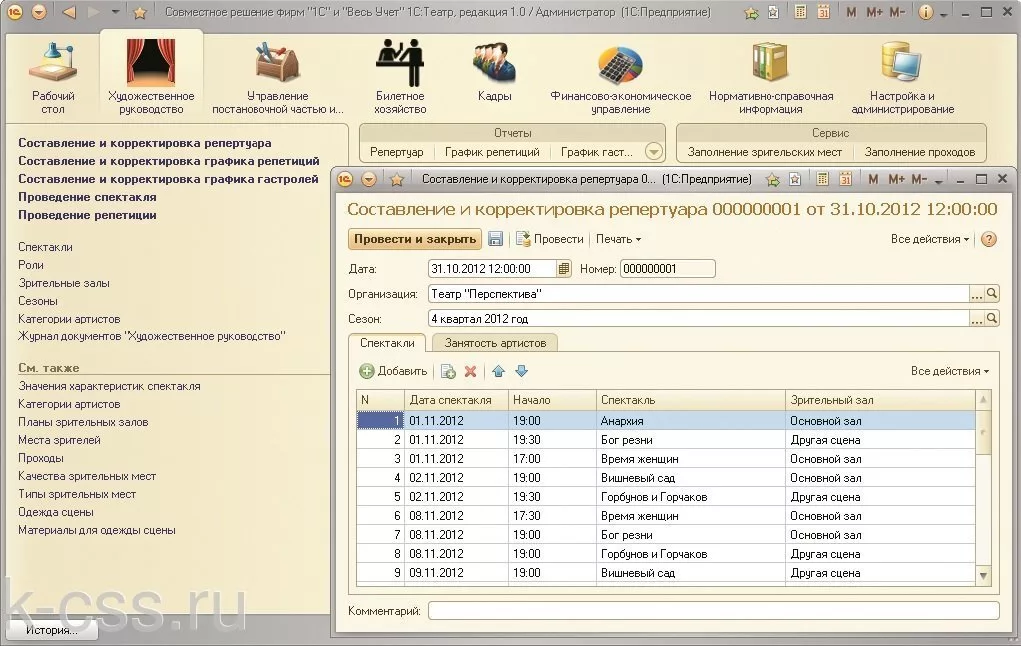


Рисунок 2 – «1C: Театр"

# **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Техническое задание (ТЗ) является документом, который содержит детальное описание требований, целей, характеристик и функциональности проекта или задачи. В процессе разработки оно играет ключевую роль, позволяя определить цели проекта, его функциональные и нефункциональные требования, а также ожидания заказчика или команды, работающей над проектом.

В ТЗ включается информация о основных целях проекта, процессе разработки, предметной области, требованиях к характеристикам, обеспечению и интерфейсу, а также описание основных этапов работы и критериев успешного завершения проекта. Кроме того, в нем могут быть указаны сроки выполнения и ограничения. Содержание и структуру технического задания на создание автоматизированных систем регулирует ГОСТ 34.602-89 "Информационная технология. Комплекс стандартов на автоматизированные системы".

Важность ТЗ заключается в том, что оно является основой для общего понимания всей командой по поводу того, что должно быть выполнено, и устанавливает рамки проекта, исключая недопонимания и несоответствия ожиданиям заказчика. Такой подход помогает сориентировать разработчиков на конечную цель и обеспечивает более эффективное выполнение задачи или проекта.

# **РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

## Моделирование ИС

### Разработка IDEF0

IDEF0 (Integrated Definition for Function Modeling) – это методология моделирования функций, используемая для анализа и проектирования бизнес-процессов.

IDEF0 представляет собой формальный язык и графическую нотацию для описания функциональных аспектов системы. Методология IDEF0 использует блок-схемы, называемые функциональными моделями, для представления иерархической структуры бизнес-процессов. Эти модели описывают функции системы, их взаимодействие, потоки данных, контрольные точки и другие аспекты процесса

На рисунке 3 изображена функциональная модель информационной системы для кинотеатра.



Рисунок 3 – контекстная диаграмма информационной системы для кинотеатра

Внешней входящей информацией являются:

* Зал
* Фильмы
* Покупатели

Управляющим воздействием будут являться:

* План по часам
* График показа фильмов

Механизмом управления будут являться:

* Диспетчер расписаний
* Администратор.
* Кассир

Внешней исходящей информацией будет являться:

* Расписание с фильмами и забронированными местами.

Диаграмма декомпозиции, предоставляет контекстную функцию, состоящую из 3 более простых функций.

На рисунке 4 представлена декомпозиция процесса создания расписания фильмов.

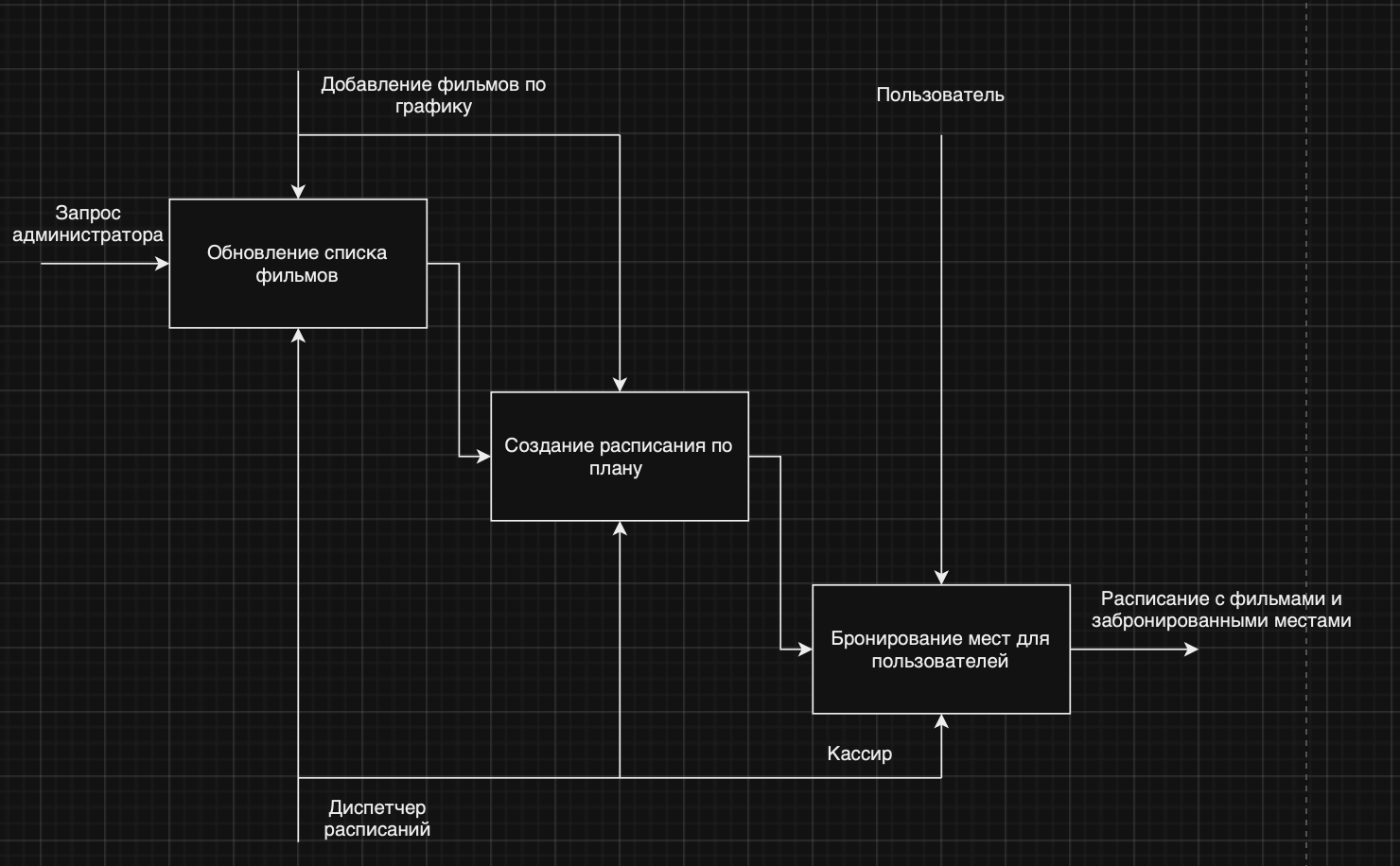


Рисунок 4 – декомпозиция процесса «Оформление заказа»

### Разработка DFD

DFD (Data Flow Diagram) – это графическое представление потоков данных в системе, показывающее, как информация перемещается через различные процессы. DFD используется для моделирования и анализа информационных систем, позволяя представить их структуру, функции и потоки данных между различными компонентами.

На рисунке 5 представлена диаграмма декомпозиции в нотации DFD работы информационной системы для кинотеатра.

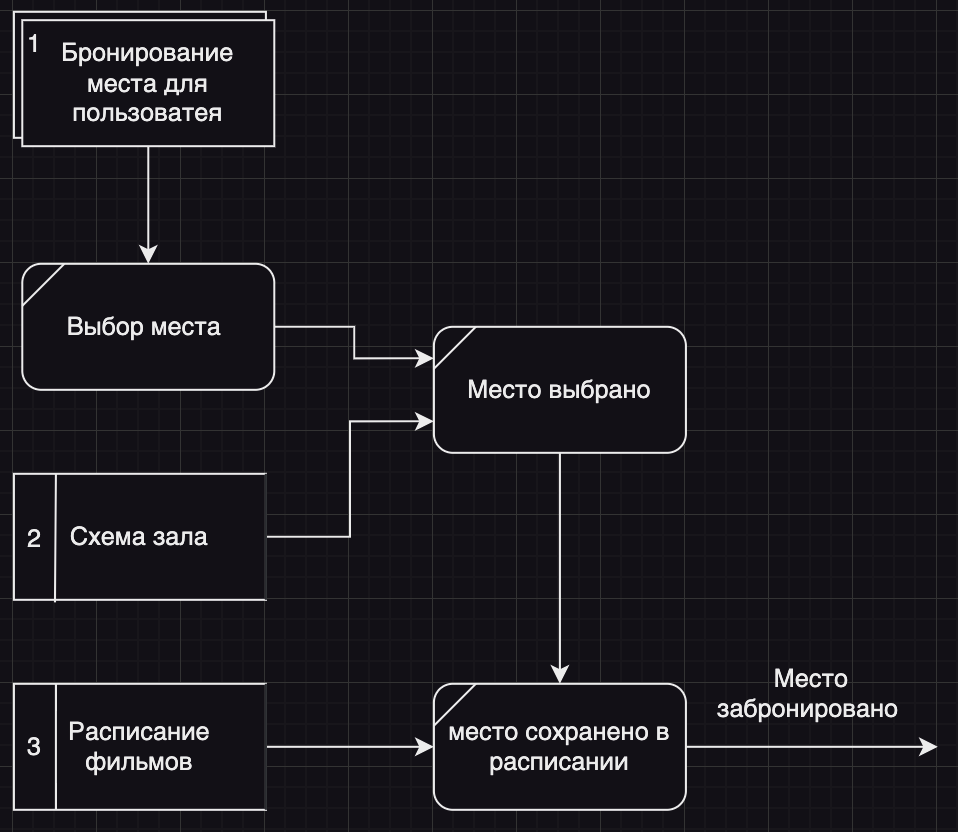


Рисунок 5 – диаграмма декомпозиции в нотации DFD ИС для кинотеатра

На диаграмме изображен процесс бронирования места: в первую очередь кассир выбирает названое место покупателем в схеме зала, далее данные сохраняются в таблице расписания и место считается забронированным.

Диаграмма содержит в себе следующие блоки:

Выбор места: выбор места в схеме зала;

Место выбрано: Место зафиксировано красным цветом в схеме зала

Расписание фильмов: данные фиксируются в базе данных связанную с позицией в таблице.

Место сохранено в расписании: место сохраняется в расписании и считается забронированным.

### Разработка диаграммы вариантов использования

Действующими лицами ИС являются:

* Кассир (бронирование места)
* Диспетчер расписания (обновление списка фильмов, изменение расписания
* БД (хранение данных о бронировании места, расписания).

Описание функциональных требований выполняют с помощью диаграммы прецедентов. Исходя из потребностей действующих лиц, выделены три варианта использования:

* + 1. Бронирование места на схеме
    2. Заполнение расписания фильмами
    3. Заполнение списка фильмов

Диаграмма прецедентов приведена на рисунке 6.

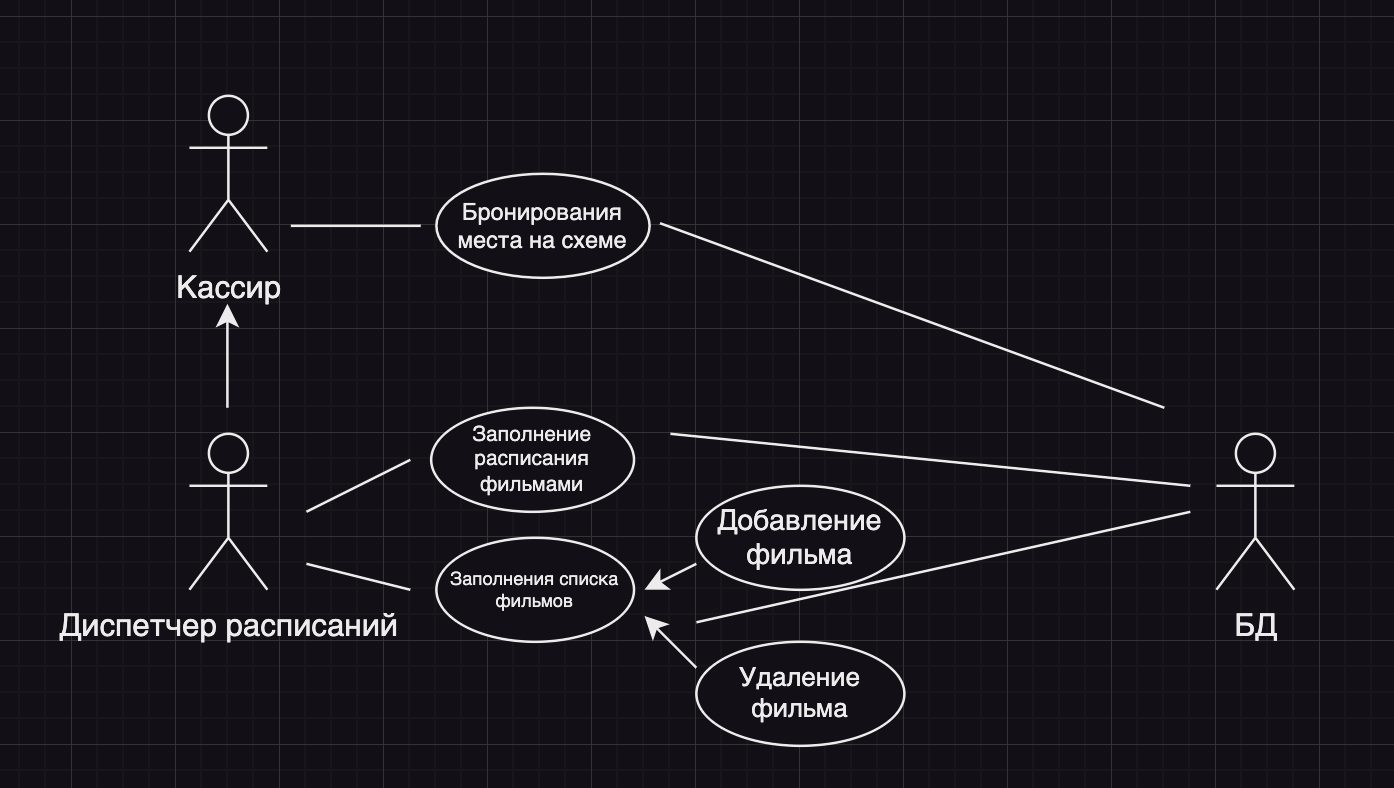


Рисунок 6 – диаграмма прецедентов ИС кинотеарта

Альтернативным вариантом является создание отдельных диаграмм прецедентов для каждого из сотрудников. На рисунке 7 представлены преценденты Кассира.



Рисунок 7 – преценденты Кассира

На рисунке 8 представлены преценденты Диспетчера расписаний



Рисунок 8 – преценденты Диспетчера расписаний

## Прототипирование и разработка интерфейса

Прототипы экранных форм представлены в Приложении А «Техническое задание» в пункте 5.2 Требования к пользовательскому интерфейсу.

Главное меню с сеткой показов представлено на рисунке 9

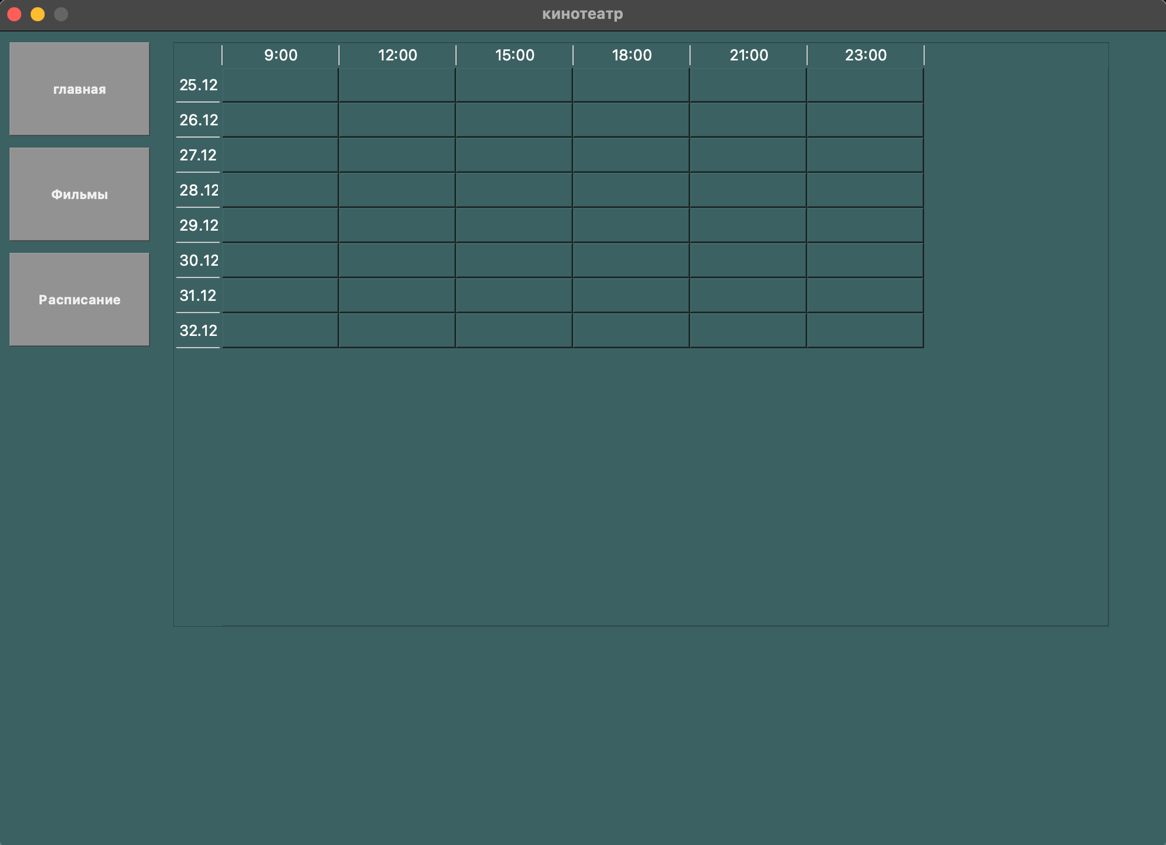


Рисунок 9 – Сетка показов

На данном рисунке предоставлен вид главного экрана, на котором можно посмотреть расписание фильмов, и при нажатии на ячейку появится окно, в котором можно будет забронировать место см. рисунок 10. На главном меню в левой части экрана присутствуют вкладки «Главная» (это уже выбранная страница), «Расписание», «Фильмы». На данной странице представлено расписание фильмов в табличном виде, после заполнения таблицы на странице «Расписание» в пустых ячейках появятся названия фильмов.

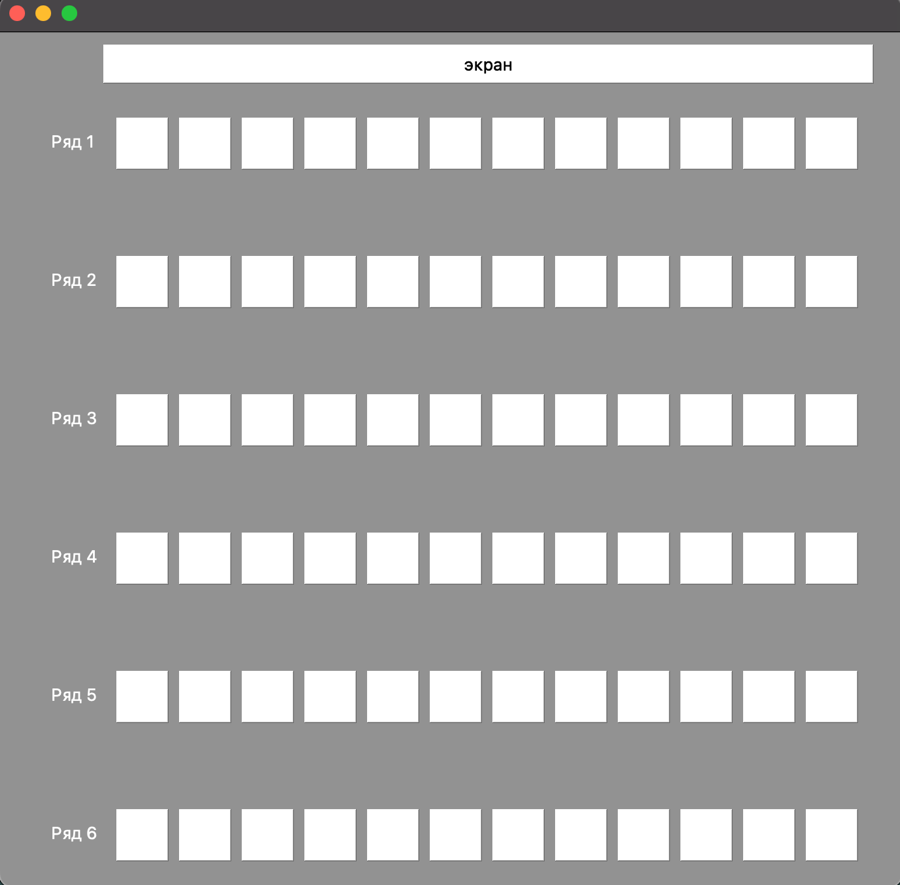


Рисунок 10 – выбор места

Нажав в главном меню на кнопку фильмы откроется окно со списком фильмов см. рисунок 11 на котором можно добавить фильм путём нажатия на кнопку «+» или убрать фильм выбрав ненужный фильм и нажав на кнопку «-»

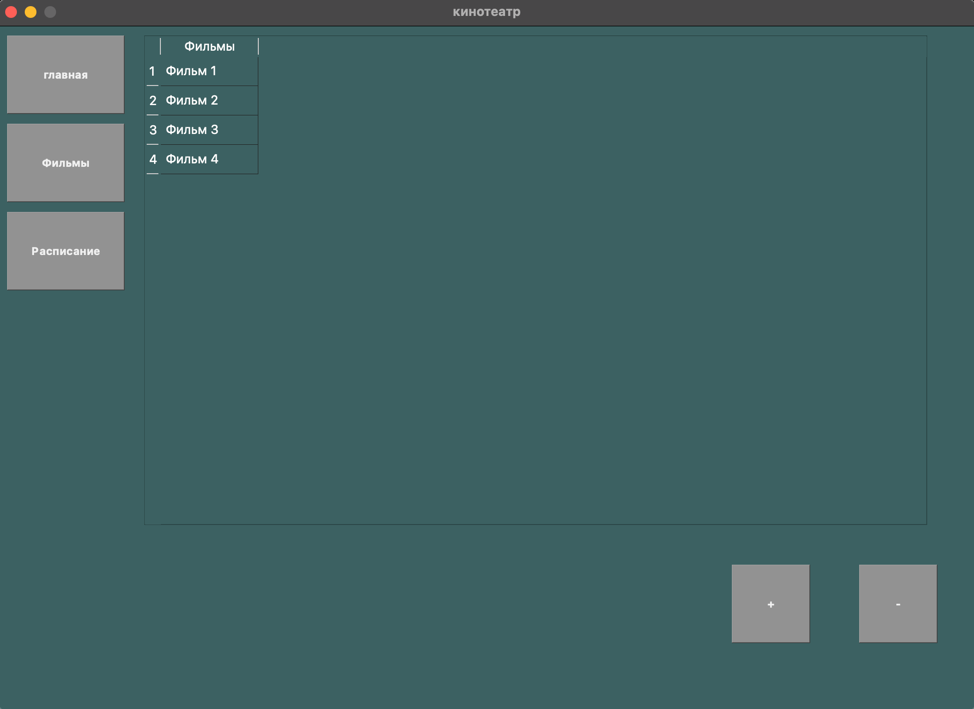


Рисунок 11 – Фильмы

Нажав в главном меню на кнопку фильмы откроется окно для заполнения расписания см. рисунок 12. После нажатия на любую из ячеек высветится всплывающее окно с выпадающим списком, в котором будут выведены объекты взятые из вкладки «Фильмы» путём их выбора из выпадающего списка.

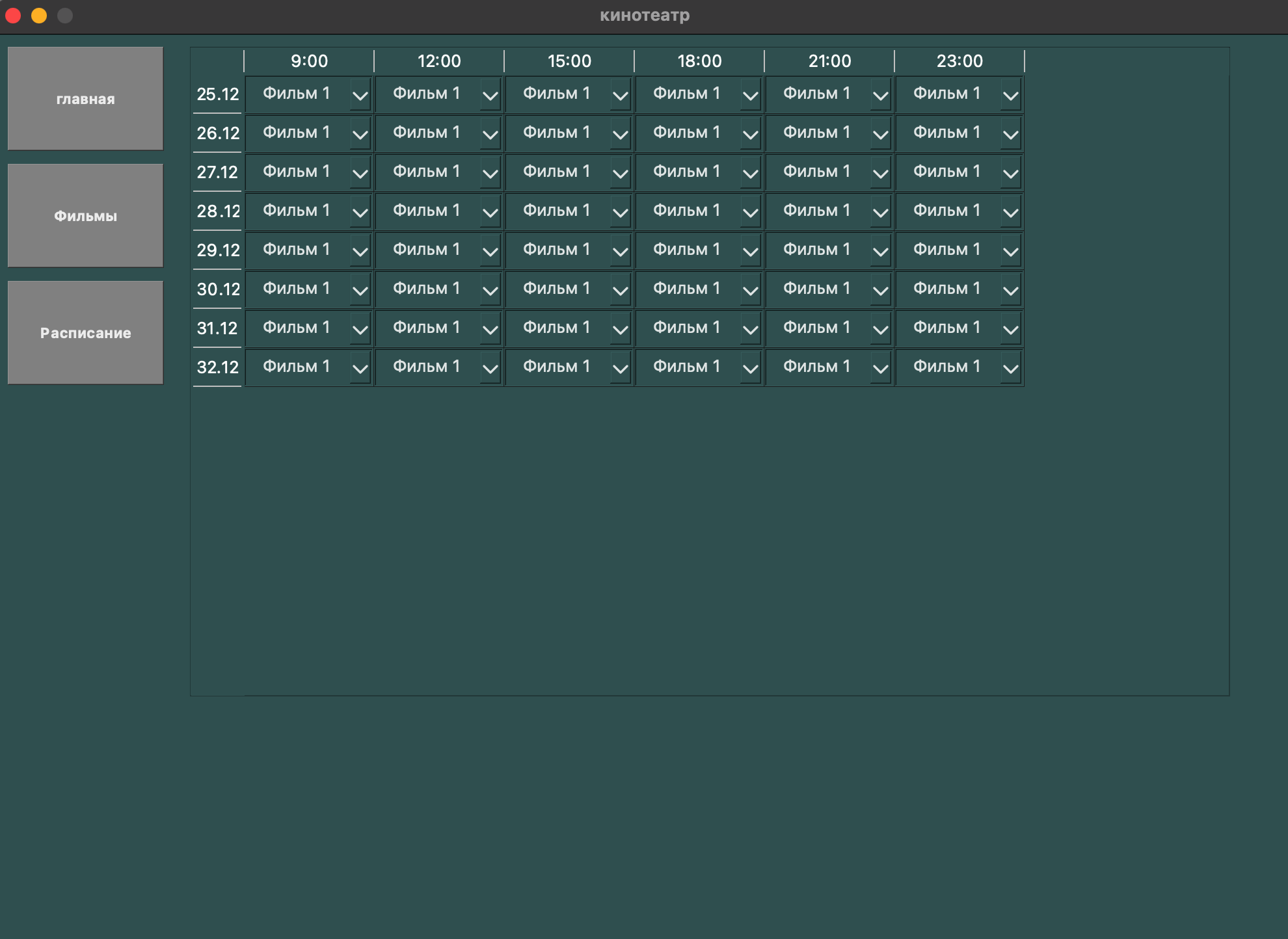


Рисунок 12 – Расписание

## Разработка базы данных

### Разработка логической и физической моделей подсистемы

Физическая модель базы данных описывает весь необходимый функционал для конкретной системы управления базами данных, включая структуру таблиц, их поля, типы данных и связи между ними через первичные и внешние ключи. Эта модель разрабатывается из логической модели с учетом требований и ограничений, присущих выбранной СУБД.

На рисунке 13 представлена физическая модель подсистемы, описывающая 4 таблицы, которые между собой имеют связи.

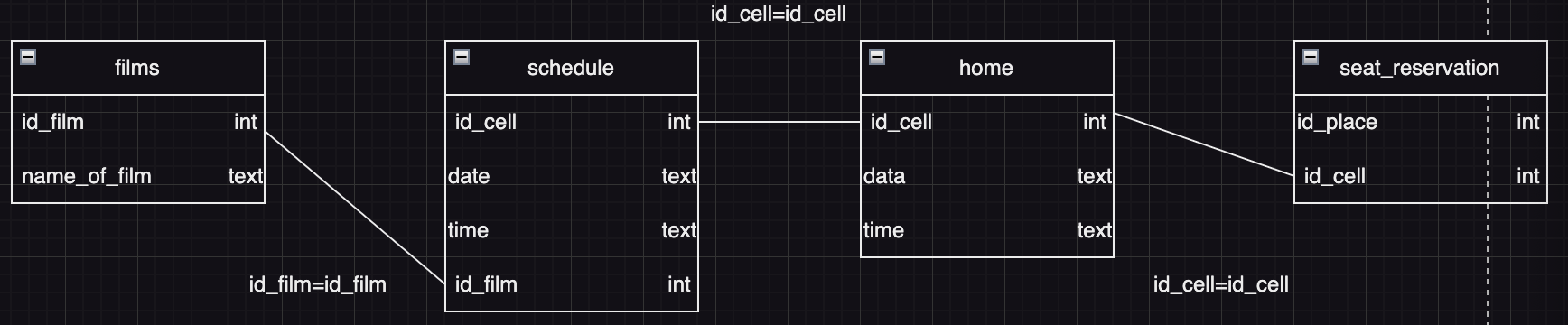


Рисунок 13 – физическая модель программного обеспечения

Данная модель имеет 4 сущности: films, schedule, home, seat\_reservation. Первичный ключ category\_id в таблице «films» связан с внешним ключом id\_category таблицы «schedule». Первичный ключ id\_cell в таблице «schedule» связан с первичным ключом «schedule» таблицы «home». Также первичный ключ id\_cell связан c внешним ключом таблицы «seat\_reservation».

Логическая модель по нотации IDEF1X помогает визуализировать структуру данных, устанавливать связи между различными элементами и обеспечивать базу данных с определенными правилами целостности.

Логическая модель по нотации IDEF1X представлена на рисунке 14.

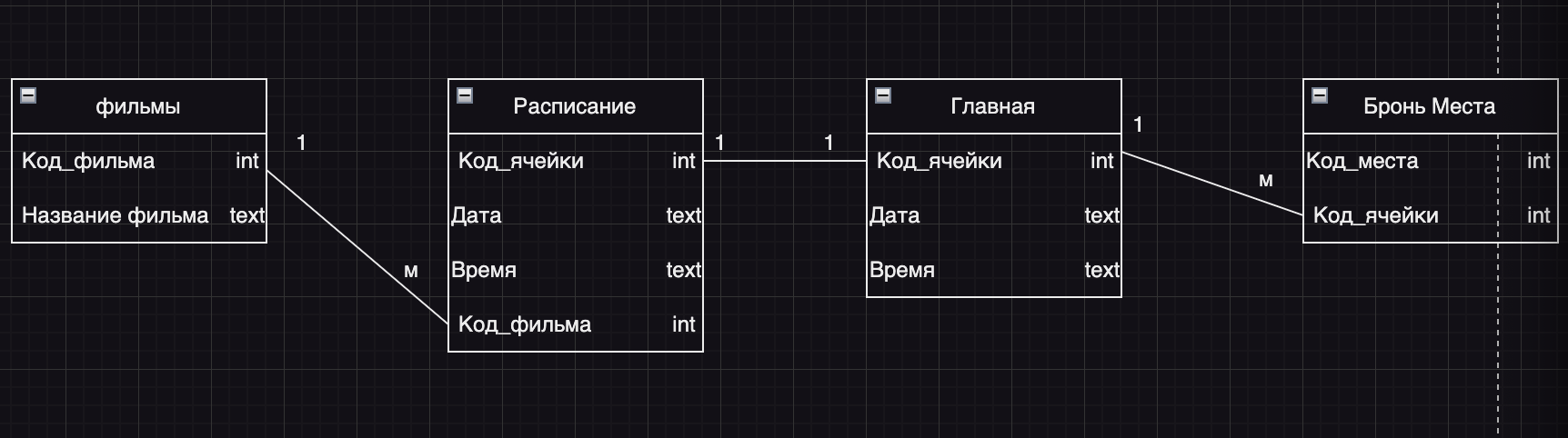


Рисунок 14 – логическая модель программного обеспечения

Данная модель имеет четыре сущности: фильмы, расписание, главная, Бронь места.

В базе данных были определены следующие связи:

1. «код\_фильма» в таблице «фильмы» связана с «код\_фильма» в таблице «Расписание» - один ко многим, один фильм может быть в нескольких ячейках.
2. «код\_ячейки» в таблице «Расписание» связана с «код\_ячейки» в таблице «Главная» - один к одному, в одной ячейке таблицы главная может храниться может храниться значение той же ячейки что и в таблице Расписание.
3. «код\_ячейки» в таблице «главная» связана с «код\_ячейки» в таблице «Бронь места» - один к одному, на одну ячейку таблицы может быть создана только одна база с местами.

# **ЗАКЛЮЧЕНИЕ**

В процессе написания курсовой работы был проведен анализ проблем, сформулирован список задач и изучена работа аналогичных систем, что привело к их успешному решению. Основной целью работы было изучение процесса создания технического задания для информационной системы.

В ходе выполнения работы были выполнены следующие этапы: проведен обзор предметной области, составлено техническое задание, разработаны решения и концепция проекта, удовлетворены все требования, спроектирован прототип пользовательского интерфейса с описанием, созданы диаграммы IDEF0 и DFD, разработаны логическая и физическая модели данных.

Таким образом, разработанное техническое задание и программное обеспечение имеют потенциал значительно улучшить и ускорить работу сотрудников, занятых в данной сфере.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ГОСТ 19.201-78 [Электронный ресурс] – URL: https://ohranatruda.ru/ot\_biblio/norma/220180/ (дата обращения: 24.11.2023).
2. Лутц. М. Изучаем Python: учебное пособие / Марк Лутц. — 5-е издание, исправленное и дополненное. – Москва: ООО Издательство «ДМК Пресс», 2020. 1216 c- ISBN: 978-5-907144-52-15.
3. Даг, Хеллман Стандартная библиотека Python 3. Справочник с примерами / Хеллман Даг. — 2. — Москва : Диалектика, 2020. — 1376 c. — Текст : непосредственный.
4. Хабр Q&A — вопросы и ответы для IT-специалистов : [сайт]. — URL: https://qna.habr.com/ (дата обращения: 30.11.2023).
5. Stackoverflow : [сайт]. — URL: <https://ru.stackoverflow.com/> (дата обращения: 30.11.2023).
6. CyberForum : [сайт]. — URL: <https://www.cyberforum.ru/> (дата обращения: 30.11.2023).
7. GitHub : [сайт]. — URL: <https://github.com/> (дата обращения: 30.11.2023).
8. ГОСТ 34.602-89 [Электронный ресурс] – URL: [ГОСТ\_34-602-89](https://irkobl.ru/sites/saio/rgistp/td/%D0%93%D0%9E%D0%A1%D0%A2_34-602-89_%D0%A2%D0%97_%D0%BD%D0%B0_%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5_%D0%90%D0%A1.pdf%23:~:text=%D0%93%D0%9E%D0%A1%D0%A2%2034.602-89.%20%D0%94%D0%B0%D1%82%D0%B0%20%D0%B2%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D1%8F%2001.01.90.,(%D0%B4%D0%B0%D0%BB%D0%B5%D0%B5%20-%20%D0%A2%D0%97%20%D0%BD%D0%B0%20%D0%90%D0%A1)%20) (24.11.2023)

# **ПРИЛОЖЕНИЕ А**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

# **ПРИЛОЖЕНИЕ Б**

**ИСХОДНЫЙ КОД**

import sys  
from datetime import datetime  
from PyQt5.QtWidgets import QApplication, QLabel, QMainWindow, QWidget, QVBoxLayout, QPushButton, QTableWidget, \  
 QTableWidgetItem, QComboBox, QGridLayout, QHBoxLayout  
  
i = 3  
z = 0  
id\_values = list(range(1, 49))  
values = []  
films = ['Фильм 1', 'Фильм 2', 'Фильм 3', 'Фильм 4']  
t1 = datetime.now().strftime('%d.%m')  
t2 = str(int(datetime.now().strftime('%d')) + 1) + str(datetime.now().strftime('.%m'))  
t3 = str(int(datetime.now().strftime('%d')) + 2) + str(datetime.now().strftime('.%m'))  
t4 = str(int(datetime.now().strftime('%d')) + 3) + str(datetime.now().strftime('.%m'))  
t5 = str(int(datetime.now().strftime('%d')) + 4) + str(datetime.now().strftime('.%m'))  
t6 = str(int(datetime.now().strftime('%d')) + 5) + str(datetime.now().strftime('.%m'))  
t7 = str(int(datetime.now().strftime('%d')) + 6) + str(datetime.now().strftime('.%m'))  
t8 = str(int(datetime.now().strftime('%d')) + 7) + str(datetime.now().strftime('.%m'))  
  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 # Создаём окно  
 self.resize(1000, 700)  
 self.setMinimumSize(1000, 700)  
 self.setMaximumSize(1000, 700)  
 self.setGeometry(400, 200, 1000, 700)  
 self.setWindowTitle('кинотеатр')  
 self.setStyleSheet("background-color: darkslategrey; ")  
  
 # Кнопка Главная  
 self.button1 = QPushButton("главная", self)  
 self.button1.setGeometry(10, 10, 120, 80)  
 self.button1.setStyleSheet("background-color: grey; ")  
 self.button1.clicked.connect(lambda: self.prem(1))  
 font = self.button1.font()  
 font.setPointSize(11)  
 font.setBold(True)  
 font.setWeight(75)  
 self.button1.setFont(font)  
 self.button1.setObjectName("button1")  
 self.button1.clicked.connect(self.clear\_table)  
 self.button1.clicked.connect(self.delete\_button)  
 self.button1.clicked.connect(self.widgets\_for\_window)  
  
 # Кнопка Фильмы  
 self.button2 = QPushButton("Фильмы", self)  
 self.button2.setGeometry(10, 100, 120, 80)  
 self.button2.setStyleSheet("background-color: grey; ")  
 self.button2.clicked.connect(lambda: self.prem(2))  
 font = self.button2.font()  
 font.setPointSize(11)  
 font.setBold(True)  
 font.setWeight(75)  
 self.button2.setFont(font)  
 self.button2.setObjectName("button2")  
 self.button2.clicked.connect(self.clear\_table)  
 self.button2.clicked.connect(self.widgets\_for\_window)  
  
 # Кнопка Расписание  
 self.button3 = QPushButton("Расписание", self)  
 self.button3.setGeometry(10, 190, 120, 80)  
 self.button3.setStyleSheet("background-color: grey; ")  
 self.button3.clicked.connect(lambda: self.prem(3))  
 font = self.button3.font()  
 font.setPointSize(11)  
 font.setBold(True)  
 font.setWeight(75)  
 self.button3.setFont(font)  
 self.button3.setObjectName("button3")  
 self.button3.clicked.connect(self.clear\_table)  
 self.button3.clicked.connect(self.widgets\_for\_window)  
 self.button3.clicked.connect(self.delete\_button)  
  
 self.table\_widget = QTableWidget(self)  
 self.table\_widget.setGeometry(150, 10, 800, 500)  
 self.widgets\_for\_window()  
  
 def prem(self, namei):  
 global i  
 i = namei  
 print(i)  
  
 def add\_combobox(self):  
 global films  
  
 for row in range(self.table\_widget.rowCount()):  
 for column in range(self.table\_widget.columnCount()):  
 combo\_box = QComboBox()  
 for film in films:  
 combo\_box.addItem(film)  
 self.table\_widget.setCellWidget(row, column, combo\_box)  
  
 # Окна для кнопок  
 def widgets\_for\_window(self):  
 global values  
 if i == 1:  
 # values = [['1', '2', '3', '4', '5', '6'],  
 # ['7', '8', '9', '10', '11', '12'],  
 # ['13', '14', '15', '16', '17', '18'],  
 # ['19', '20', '21', '22', '23', '24'],  
 # ['25', '26', '27', '28', '29', '30'],  
 # ['31', '32', '33', '34', '35', '36'],  
 # ['37', '38', '39', '40', '41', '42'],  
 # ['43', '44', '45', '46', '47', '48']]  
 # устанавливаем количество строк и столбцов в таблице  
 self.table\_widget.setRowCount(8)  
 self.table\_widget.setColumnCount(6)  
  
 # устанавливаем заголовки для столбцов и строк  
 self.table\_widget.setHorizontalHeaderLabels(['9:00', '12:00', '15:00', '18:00', '21:00', '23:00'])  
 self.table\_widget.setVerticalHeaderLabels([str(t1), t2, t3, t4, t5, t6, t7, t8])  
 self.insertInTable(values)  
  
  
 if i == 2:  
 self.create\_list()  
  
 if i == 3:  
 #Таблица 2  
 self.table\_widget.setRowCount(8)  
 self.table\_widget.setColumnCount(6)  
 self.table\_widget.setHorizontalHeaderLabels(['9:00', '12:00', '15:00', '18:00', '21:00', '23:00'])  
 self.table\_widget.setVerticalHeaderLabels([str(t1), t2, t3, t4, t5, t6, t7, t8])  
  
 self.add\_combobox()  
  
 values = []  
 for row in range(self.table\_widget.rowCount()):  
 row\_items = []  
 for column in range(self.table\_widget.columnCount()):  
 item = self.table\_widget.item(row, column)  
 if item is not None:  
 row\_items.append(item.text())  
 else:  
 row\_items.append("")  
 values.append(row\_items)  
  
 def insertInTable(self, values):  
 global id\_values  
 for row in range(len(values)):  
 for column in range(len(values[row])):  
 button = QPushButton()  
 button.setText(str(values[row][column]))  
 id\_values = values[row][column]  
 button.clicked.connect(lambda checked, id\_values=id\_values: self.openWindow(id\_values))  
 self.table\_widget.setCellWidget(row, column, button)  
 print(id\_values)  
  
 def openWindow(self, id\_value):  
 print("a")  
 self.openw = rules()  
  
 def deleteSelectedRow(self):  
 index = self.table\_widget.currentIndex()  
 self.table\_widget.removeRow(index.row())  
 self.table\_widget.update()  
  
 def add\_button(self):  
 # Кнопка Добавить ячейку  
 global z  
 if z == 0:  
 self.button4 = QPushButton("+", self)  
 self.button4.setGeometry(750, 550, 80, 80)  
 self.button4.setStyleSheet("background-color: grey; ")  
 self.button4.clicked.connect(self.add\_cell)  
 font = self.button4.font()  
 font.setPointSize(11)  
 font.setBold(True)  
 font.setWeight(75)  
 self.button4.setFont(font)  
 self.button4.setObjectName("button4")  
 self.button4.show()  
  
 # Кнопка удалить ячейку  
  
 self.button5 = QPushButton("-", self)  
 self.button5.setGeometry(880, 550, 80, 80)  
 self.button5.setStyleSheet("background-color: grey; ")  
 self.button5.clicked.connect(self.deleteSelectedRow)  
 font = self.button5.font()  
 font.setPointSize(11)  
 font.setBold(True)  
 font.setWeight(75)  
 self.button5.setFont(font)  
 self.button5.setObjectName("button4")  
 self.button5.show()  
  
 z = 1  
  
 def delete\_button(self):  
 global z  
 if z == 1:  
 self.button4.deleteLater()  
 self.button5.deleteLater()  
 z = 0  
 print('delok')  
  
 def create\_list(self):  
 global films  
 self.table\_widget.setRowCount(0)  
 self.table\_widget.setColumnCount(1)  
 self.table\_widget.setHorizontalHeaderLabels(['Фильмы'])  
  
 #добавление элементов в список  
  
 for i, film in enumerate(films):  
 item = QTableWidgetItem(film)  
 self.table\_widget.insertRow(i)  
 self.table\_widget.setItem(i, 0, item)  
  
 # Создание кнопки "Добавить ячейку"  
 self.add\_button()  
  
 # Очистка table\_widget  
 def clear\_table(self):  
 sender\_button = self.sender()  
 if sender\_button != self.button2:  
 self.table\_widget.setRowCount(0)  
 self.table\_widget.setColumnCount(0)  
  
 # Добавление ячейки в таблицу на вкладке "Фильмы"  
 def add\_cell(self):  
 row\_count = self.table\_widget.rowCount()  
 self.table\_widget.insertRow(row\_count)  
 item = QTableWidgetItem('')  
 self.table\_widget.setItem(row\_count, 0, item)  
  
class rules(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setGeometry(630, 315, 640, 480)  
 self.setStyleSheet("background-color: grey; ")  
  
 self.button = QPushButton("экран", self)  
 self.button.setGeometry(80, 10, 590, 30)  
 self.button.setStyleSheet("background-color: white; color: black")  
  
 # Создаем вертикальный лэйаут для всех рядов кнопок и надписей  
 v\_layout = QVBoxLayout()  
 # Устанавливаем интервал между рядами  
 v\_layout.setSpacing(20)  
  
 # Создаем шесть горизонтальных лэйаутов для кнопок и надписей  
 for i in range(6):  
 h\_layout = QHBoxLayout()  
 # Устанавливаем интервал между кнопками  
 h\_layout.setSpacing(20)  
 # Устанавливаем отступы по бокам и сверху  
 h\_layout.setContentsMargins(20, 50, 20, 0)  
  
 # Добавляем надпись с номером ряда  
 label = QLabel(f"Ряд {i+1}")  
 h\_layout.addWidget(label)  
  
 # Создаем 12 кнопок и добавляем их в горизонтальный лэйаут  
 for j in range(12):  
 button = QPushButton(str(j+1))  
 button.setFixedSize(40, 40)  
 button.setStyleSheet("background-color: white; ")  
 button.clicked.connect(lambda checked, button=button: self.on\_button\_click(button))  
 h\_layout.addWidget(button)  
  
 # Добавляем горизонтальный лэйаут в вертикальный  
 v\_layout.addLayout(h\_layout)  
  
 self.setLayout(v\_layout)  
 self.show()  
  
 # Метод для изменения цвета кнопки при нажатии на нее  
 def on\_button\_click(self, button):  
 if button.styleSheet() == "background-color: white; ":  
 button.setStyleSheet("background-color: red; ")  
 else:  
 button.setStyleSheet("background-color: white; ")  
  
 # Метод для изменения цвета кнопки при нажатии на нее  
 def on\_button\_click(self, button):  
 if button.styleSheet() == "background-color: white; ":  
 button.setStyleSheet("background-color: red; ")  
 else:  
 button.setStyleSheet("background-color: white; ")  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MainWindow()  
 ex.show()  
 sys.exit(app.exec\_())