

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка приложения «Кинотеатр»

Выполнил студент

Лихторенко О. С.

Группа П1-17

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Проверил преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

Королёв 2020 г.

Оглавление

[Введение 3](#_Toc44170457)

[Глава 1. Теоретическая часть 4](#_Toc44170458)

[1.1 Изучение предметной области 4](#_Toc44170459)

[1.2 Изучение существующих разработок 6](#_Toc44170460)

[Глава 2. Проектная часть 8](#_Toc44170461)

[2.1 Построение диаграммы прецедентов 8](#_Toc44170462)

[2.2 Выбор инструментов 12](#_Toc44170463)

[2.3 Проектирование сценария 13](#_Toc44170464)

[2.4 Построение диаграммы классов 14](#_Toc44170465)

[2.5 Описание главного модуля 15](#_Toc44170466)

[2.6 Описание спецификаций к модулям 17](#_Toc44170467)

[2.7 Описание модулей 19](#_Toc44170468)

[2.8 Описание тестовых наборов модулей 20](#_Toc44170469)

[2.9 Описание применения средств отладки 22](#_Toc44170470)

[2.10 Анализ оптимальности использования памяти и быстродействия 23](#_Toc44170471)

[Глава 3. Эксплуатационная часть 26](#_Toc44170472)

[3.1. Руководство оператора 26](#_Toc44170473)

[3.1.2. Назначение программы 26](#_Toc44170474)

[3.1.3. Условия выполнения программы 26](#_Toc44170475)

[3.1.4. Выполнение программы 26](#_Toc44170476)

[Заключение 30](#_Toc44170477)

[Список литературы и интернет-источников 31](#_Toc44170478)

[Приложения 32](#_Toc44170479)

# **Введение**

Целью данного курсового проекта является написание программы «Кинотеатр» для кинотеатра второго экрана вида миниплекс. Данная программа позволит облегчить работу сотрудникам, главным образом букеру, путем систематизации данных о сеансах, проекторов, количества мест и учетов лицензионных ключей.

Далее будут описаны цели программы, ее применение и сама программа. Записка разделена на три главы.

В первой главе будет рассмотрена предметная область данной тем, а также приведены примеры уже существующих разработок в данной области, их описание, характеристики и отзывы пользователей.

Во второй части будут рассмотрены инструменты и модули, которые были разработаны, структура программной части и листинги ключевых частей программных модулей.

В третьей части будет рассмотрено руководство для пользователей, а также изображение графического интерфейса программы.

В результате этих трех главах будет полностью описана разработка, ее суть и назначения, а также сама область применения. Что полностью опишет всю разработку.

# **Глава 1. Теоретическая часть**

## **Изучение предметной области**

Первый кинотеатр появился в конце 1890-х годов вместе с зарождением [кинематографа](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%B5%D0%BC%D0%B0%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84).

Кинотеатры подразделяются на: круглогодичные (многозальные, однозальные) и сезонные (летние открытые и летние закрытые).

Цифровые копии, в том числе и 3D фильмы. Приходят в специальных кейсах, на переносных HDD дисках, либо по защищенной сети или спутниковый ретранслятор. Копии приходят в специальном формате DCP (Digital Cinema Package). Фильм в среднем весит от 230 до 350 гигабайт. Эта технология практически вытеснила традиционную пленочную благодаря многим преимуществам, например возможность проведения мировых премьер одновременно в разных точках планеты.

Для того, чтобы показать картину, кинотеатру необходимо получить лицензию на показ и цифровой ключ (KDM, Key Delivering Message), в котором зашифровано количество сеансов и их время. Также в ключе указываются серийные номера конкретных серверов, которыми можно показать кинокартину, а с 2009 года и номера кинопроекторов. В случае поломки сервера его замена должна согласовываться с компанией-дистрибьютором, у которой запрашивается новый ключ KDM. Та же процедура обязательна при смене расписания киносеансов, в противном случае показ будет невозможен вследствие блокировки сервера лицензионным ключом.

На данный момент существуют следующие виды кинотеатров:

* Моноэкран — кинотеатр с одним экраном;
* Миниплекс — комплекс с количеством экранов от 2 до 8;
* Мультиплекс — кинотеатр от 9 до 15 залов;
* Мегаплекс — от 16 залов;
* Киноплекс — кинотеатр, расположенный в торговом или торгово-развлекательном центре;
* Кинотеатры «первого экрана» — выпускают фильмы в день премьеры по всей стране;
* Кинотеатры «второго экрана» — прокатывают фильмы через две-три недели после премьеры на «первом экране».

## **Изучение существующих разработок**

Рассмотрим приложение для работников и гостей кинотеатра компании ООО «Киноплан». Все материалы взяты с их официального сайта.

Реализована автоматическая отчётность перед дистрибьюторами, налоговыми и государственными службами, что упрощает работу кассира и букера.

Функция планирование репертуара (рис. 1), так же облегчает работу букера и менеджера кинотеатра.

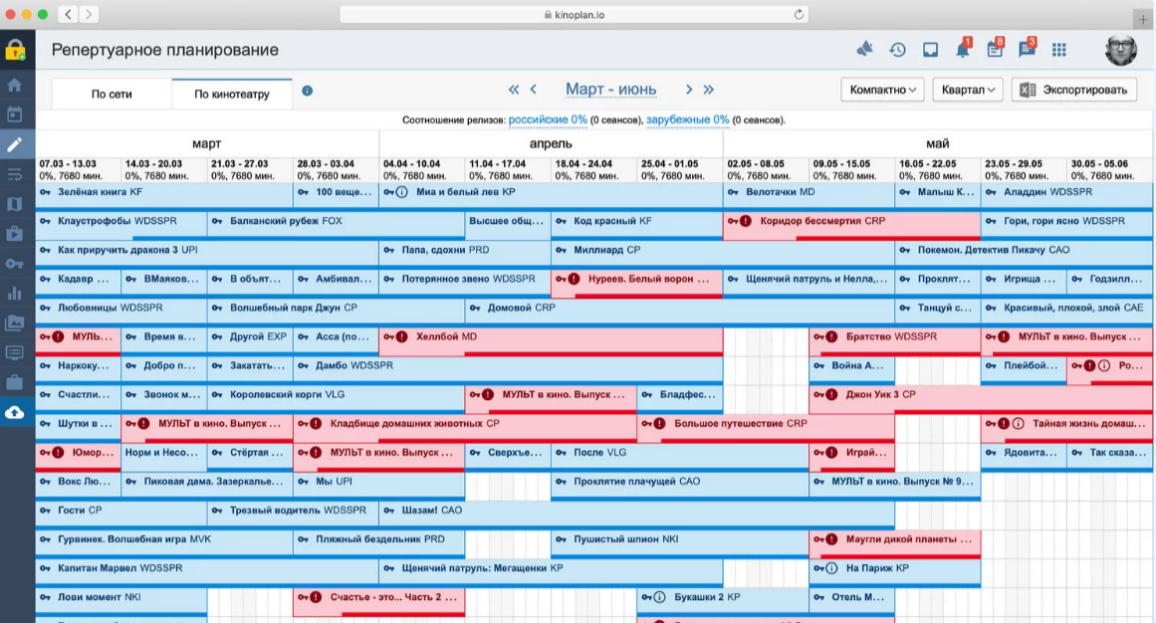


Рисунок 1. Планирование репертуара

Автоматическое расписание сеансов (рис. 2), позволяет экономить время и заранее планировать долгосрочные премьеры и показы.

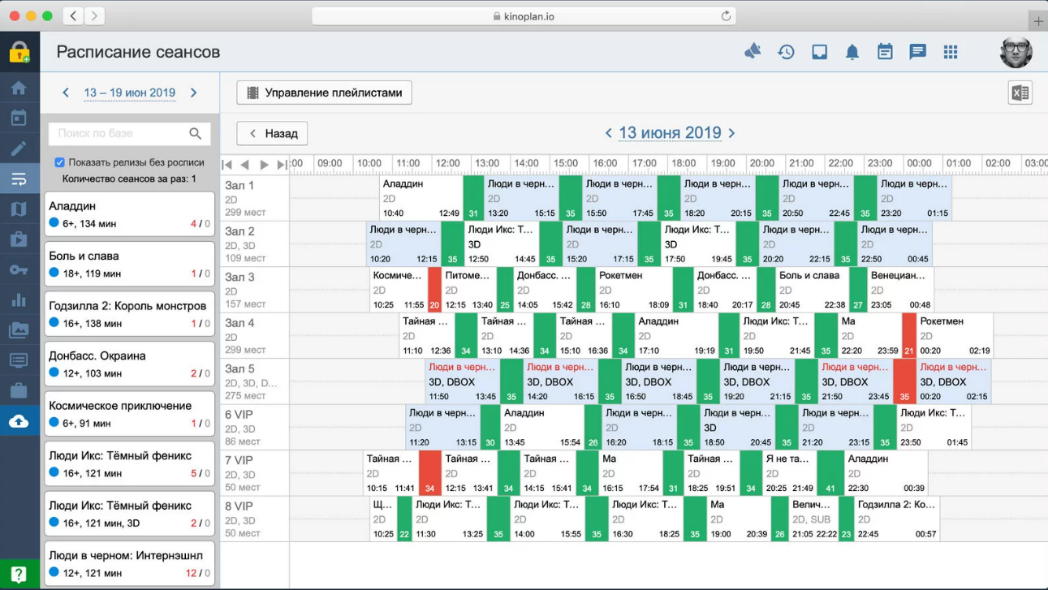


Рисунок 2. Расписание

Функция управление KDM ключами (рис. 3), позволяет сократить персонал и рабочие часы.

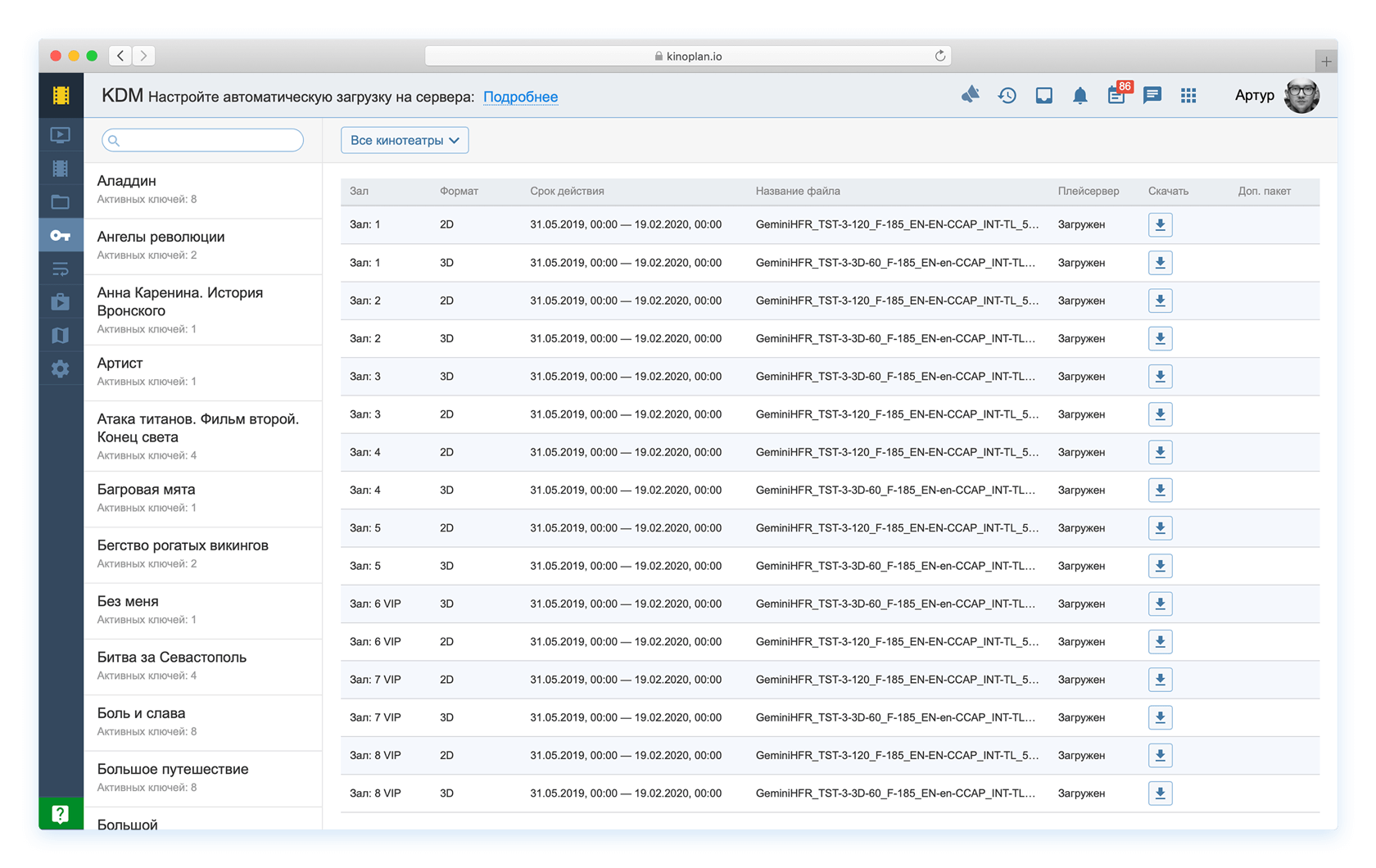


Рисунок 3. KDM ключи

# **Глава 2. Проектная часть**

## **Построение диаграммы** **прецедентов**

Были построены диаграммыпрецедентов для «Администратора», «Букера», «Киномеханика», «Кассира» и «Клиента»

На данных диаграммах мы можем увидеть какой пользователь за какое действие отвечает. Далее вы можете рассмотреть общую диаграмму прецедентов (рис. 4).

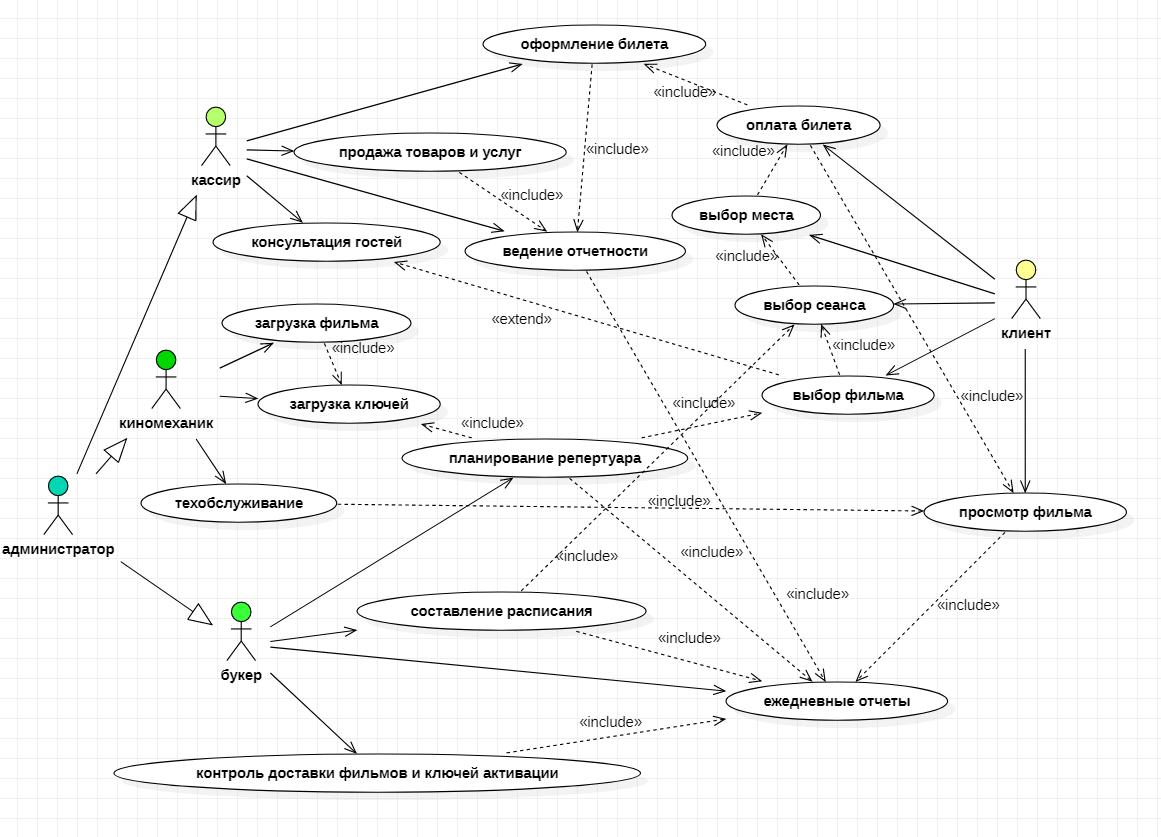


Рисунок 4. Общая диаграмма

Диаграмма прецедентов «Администратора» (рис. 5), показывает что администратор осуществляет контроль за работниками.

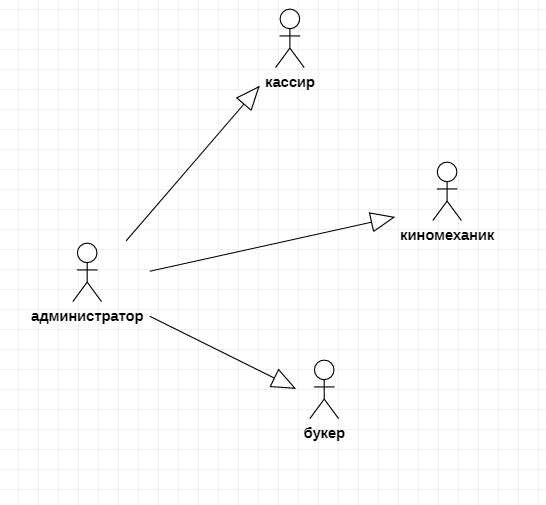


Рисунок 5. Диаграмма администратора

Диаграмма прецедентов «Кассира» (рис. 6). В обязанности входит: обслуживание клиентов, ведение отчетности проданных билетов и кассы

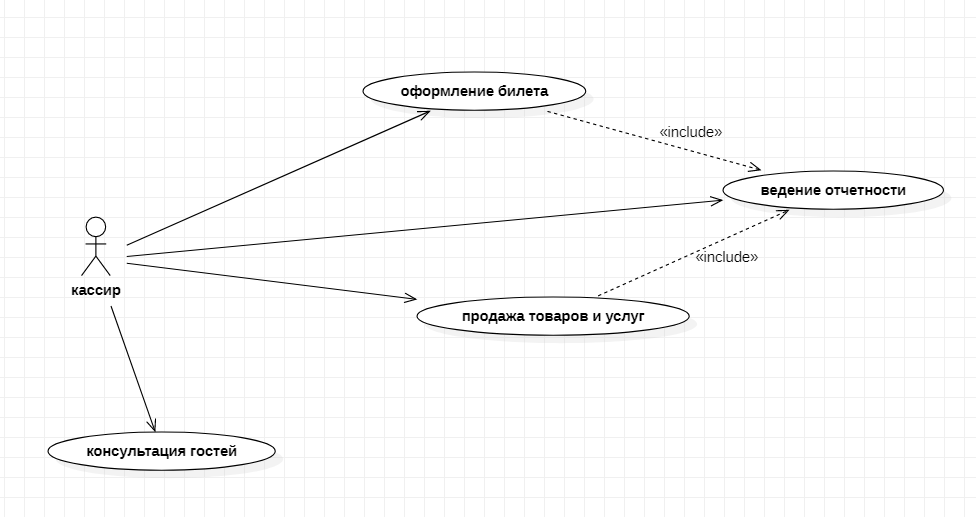


Рисунок 6. Диаграмма кассира

Диаграмма прецедентов «Киномеханик» (рис. 7) показывает, что киномеханик не ведет никакой отчетности, но обслуживает проекторы и залы

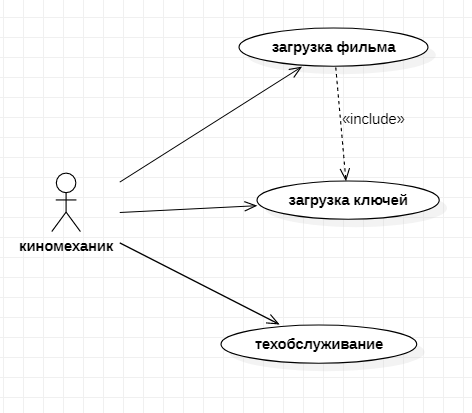


Рисунок 7. Диаграмма киномеханика

Один из самых важных работников – Букер. Как мы видим (рис. 8), кинопоказ осуществляется под его надзором. В его обязанности входит планирование репертуара и расписания кинотеатра, и, самое главное, на чем и будем делать акцент – контроль за KDM.



Рисунок 8. Диаграмма букера

Диаграмма «Клиент» (рис. 9) показывает один из сценариев поведения человека, пришедшего в кинотеатр.

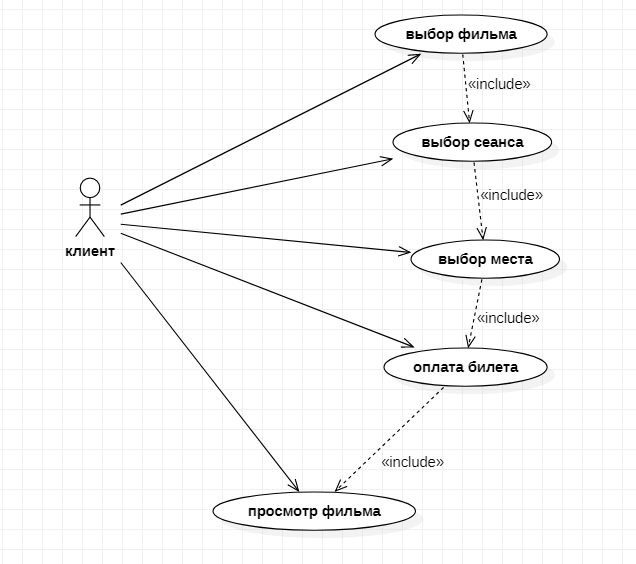


Рисунок 9. Диаграмма клиента

## **Выбор инструментов**

При выборе инструментов обращалось внимание на определенные критерии:

* Наличие необходимых библиотек
* Возможность подключений сторонних систем
* Наличие документации на русском языке
* Возможность работы на различных платформах
* Актуальность
* Доступность

После некоторых поисков был выбран язык программирования Object Pascal, среда разработки Lazarus, GUI-оболочка IBExpert и система управления базами данных Firebird.

## **Проектирование сценария**

Данная программа ориентирована на такой сценарий (рис. 10).

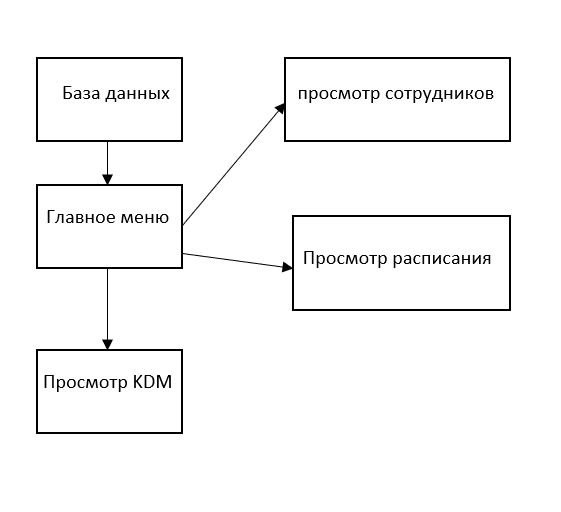


Рисунок 10. Сценарий

После запуска программы пользователь имеет несколько вариантов действий

1. Просмотр сотрудников
2. Просмотр ключей KDM с актуальной информацией
3. Просмотр расписание кинотеатра
4. Удаление, добавление и редактирование таблицы сотрудников
5. Работа с KDM
6. Работа с расписанием

При выборе просмотра любой таблицы пользователь будет получать готовую к просмотру соответствующую таблицу.

## **Построение диаграммы классов**

В данном параграфе представлены все классы, которые используются в программе, а также их взаимосвязи (рис. 11).

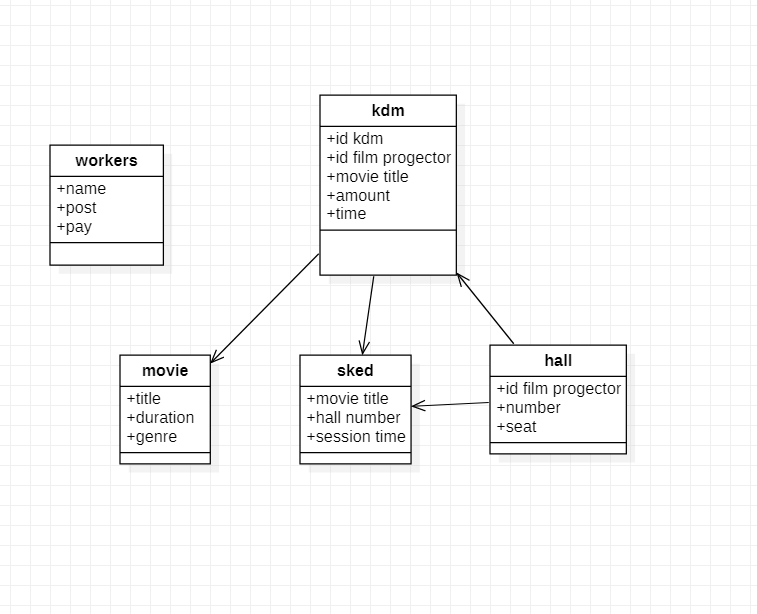


Рисунок 11. Диаграмма классов

## **Описание главного модуля**

В главный (центральный) модуль разработки входит класс Подсоединение БД, которые отвечает за взаимосвязь со всеми компонентами Базы данных, а также за их включение.

Данный модуль запускается автоматически при запуске программы что обеспечивает качественное соединение и отсутствие ошибок во время работы.

Данный модуль состоит из одного кода, который приведен в листинге 1.

**Листинг 1:**

unit umain; //заголовок модуля

{$mode objfpc}{$H+}

Interface //интерфейсная часть

uses

Classes, SysUtils, db, Forms, Controls, Graphics, Dialogs, DBGrids, DBCtrls,

Calendar, EditBtn, StdCtrls, DBExtCtrls, ExtDlgs, ComCtrls, PopupNotifier,

ExtCtrls, Buttons, Menus, IBTable, VirtualTrees, DBDateTimePicker,

DateTimePicker, RTTIGrids, RTTICtrls, Types;

type

//список элементов формы

{ TFMain }

TFMain = class(TForm)

DBGrid1: TDBGrid;

DBGrid2: TDBGrid;

DBGrid3: TDBGrid;

DBGrid4: TDBGrid;

DBNavigator1: TDBNavigator;

DBNavigator2: TDBNavigator;

DBNavigator3: TDBNavigator;

DBNavigator4: TDBNavigator;

movie: TPageControl;

TabSheet1: TTabSheet;

workers: TTabSheet;

KDM\_key: TTabSheet;

timetable: TTabSheet;

procedure cmbkdmChange(Sender: TObject);

procedure DBComboBox1Change(Sender: TObject);

procedure DBNavigator1Click(Sender: TObject; Button: TDBNavButtonType);

procedure sdfsClick(Sender: TObject);

//procedure movieChange(Sender: TObject);

procedure KDM\_keyContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

//procedure movieChange(Sender: TObject);

procedure TILabel1Click(Sender: TObject);

procedure timetableContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

procedure TIProgressBar1DragDrop(Sender, Source: TObject; X, Y: Integer);

procedure workersContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

private

public

end;

## **Описание спецификаций к модулям**

Центральный модуль представляет из себя набор обязательных элементов базы данных таких как таблицы, запросы и транзакции. И центральную функцию, которая представляет из себя запуск и настройку базы данных.

Модуль просмотра таблиц представляет из себя несколько графических форм каждая из которых обращается в центральный модуль за своим запросом из базы данных для дальнейшей его вывода, так же подключает навигацию к полученному запросу. Так же этот модуль имеет два скрипта которые представляют из себя:

* 1. Подстановку в выпадающий список соответствующе значения;
  2. При выборе значения из списка производит изменение скрипта запроса и выводит новую отсортированную таблицу.

**Листинг 2. Модуль просмотра таблиц:**

unit udm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, db, IBDatabase, IBTable, IBCustomDataSet, IBQuery, StdCtrls;

type

{ Tdm }

Tdm = class(TDataModule)

DataSource1: TDataSource;

dtsfilm: TDataSource;

dtsTimetable: TDataSource;

dtsWorkers: TDataSource;

ibdb: TIBDatabase;

ibfilmDURATION: TTimeField;

ibfilmGENRE: TIBStringField;

ibfilmNAME: TIBStringField;

ibkdm: TIBTable;

ibkdmAMOUNT: TIBIntegerField;

ibkdmID: TIBIntegerField;

ibkdmID\_FILM\_PROJECTOR: TIBIntegerField;

ibkdmNAME: TIBStringField;

ibkdmTIME: TTimeField;

ibqKDMID: TIBIntegerField;

ibqKDMID\_FILM\_PROJECTOR: TIBIntegerField;

ibqKDMNAMBER: TIBIntegerField;

ibqKDMNAME: TIBStringField;

ibqKDMSEAT: TIBIntegerField;

ibfilm: TIBTable;

ibTimetable: TIBTable;

IBTable1AMOUNT: TIBIntegerField;

IBTable1ID: TIBIntegerField;

IBTable1ID\_FILM\_PROJECTOR: TIBIntegerField;

IBTable1NAME: TIBStringField;

IBTable1TIME: TTimeField;

ibTimetableNAMBER: TIBIntegerField;

ibTimetableNAME: TIBStringField;

ibTimetableSESSIONTIME: TTimeField;

ibWorkers: TIBTable;

ibtr: TIBTransaction;

ibWorkersID: TIBIntegerField;

ibWorkersNAME: TIBStringField;

ibWorkersPAY: TIBBCDField;

ibWorkersPOST: TIBStringField;

procedure DataModuleCreate(Sender: TObject);

procedure DataSource1DataChange(Sender: TObject; Field: TField);

procedure dtsfilmDataChange(Sender: TObject; Field: TField);

procedure dtsKDMDataChange(Sender: TObject; Field: TField);

procedure dtsTimetableDataChange(Sender: TObject; Field: TField);

procedure dtsWorkersDataChange(Sender: TObject; Field: TField);

private

public

skladIndexes:TstringList;

procedure LoadTable(tableName:string;strl:TstringList;cmd:Tcombobox);

end;

var

dm: Tdm;

implementation

{$R \*.lfm}

{ Tdm }

procedure Tdm.DataModuleCreate(Sender: TObject);

begin //вручную открываем необходимые таблицы

ibtr.StartTransaction;

ibWorkers.Open;

ibTimetable.Open;

//ibqKDM.Open;

ibkdm.Open;

ibfilm.Open;

// KMDIndexes:=TstringList.Create;

// LoadTable('KDM\_key',nil,FMain.cmbkdm);

end;

## **Описание модулей**

На рисунке 12 приведен модуль данных, отвечающий за работу с БД. Он содержит в себе компонент транзакции TIBTransaction (ibtr), компонент соединения с базой данных TIBDataBase (ibdb), компоненты таблиц TIBTable (ibTimetable, ibWorkers, ibfilm, ibkdm) и компоненты TDataSource, отвечающие за управление потоками данных, за связи между набором данных и визуальными компонентами (dtsTimetable, dtsWorkers, dtsfilm, DataSource1).

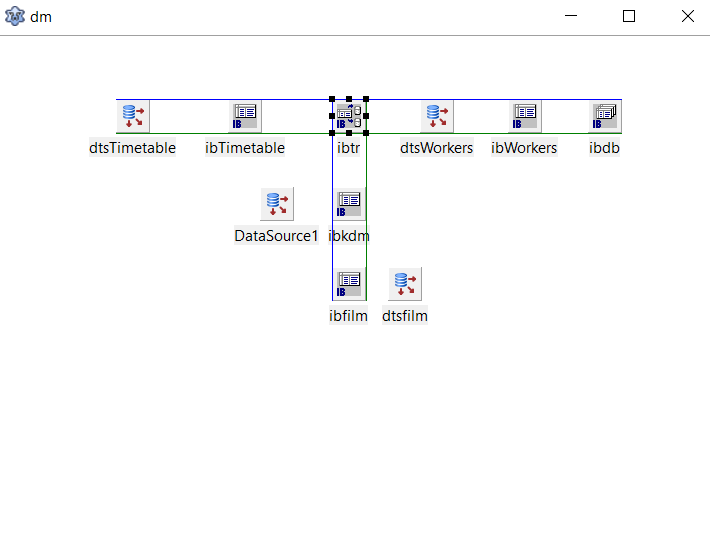


Рисунок 12. Модуль данных

## **Описание тестовых наборов модулей**

На рисунках ниже приведены результаты работы добавления записи в таблице. Рисунок 13 - до добавлении записи, рисунок 14 – после.

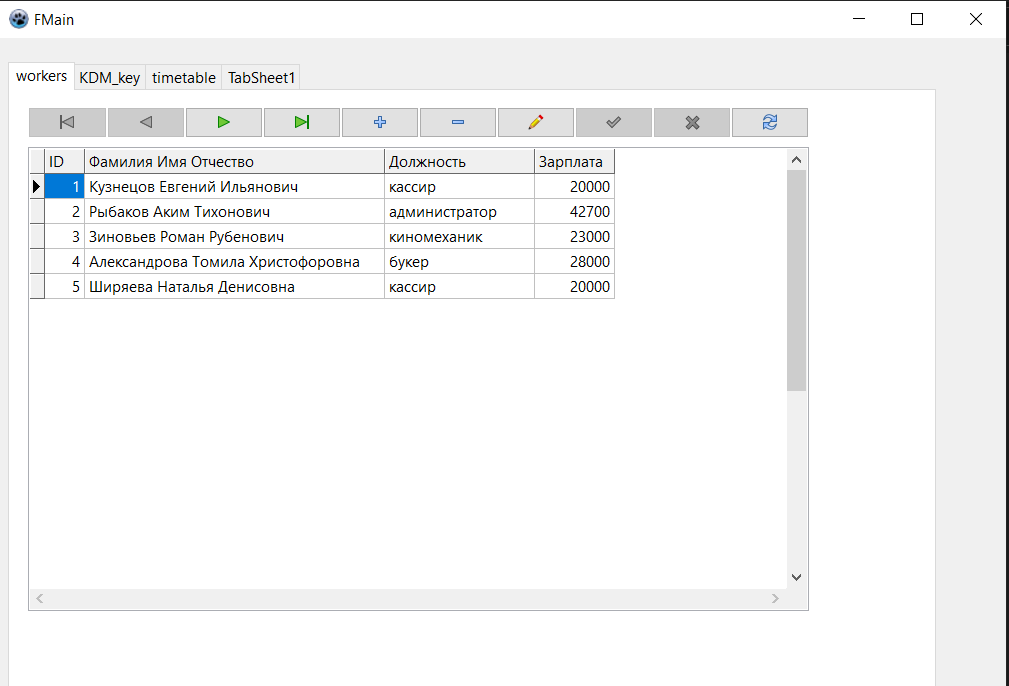


Рисунок 13. До добавления записи о сотруднике

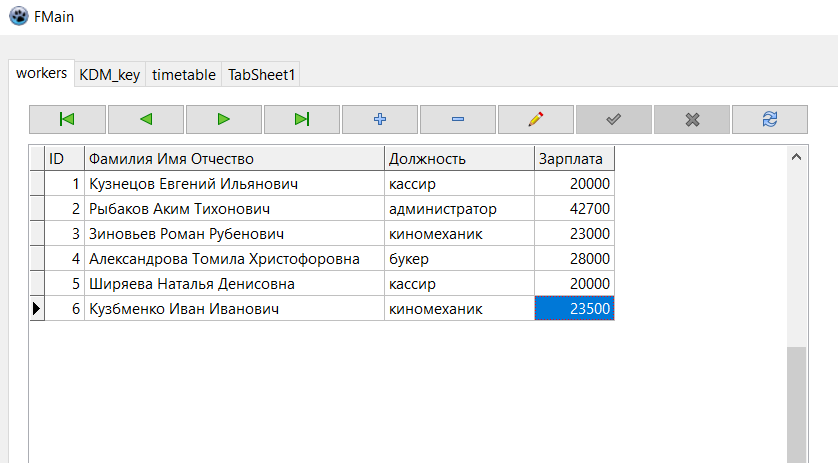


Рисунок 14. С новой записью.

Проверяем можем ли мы удалить записи (рис. 15, рис. 16).

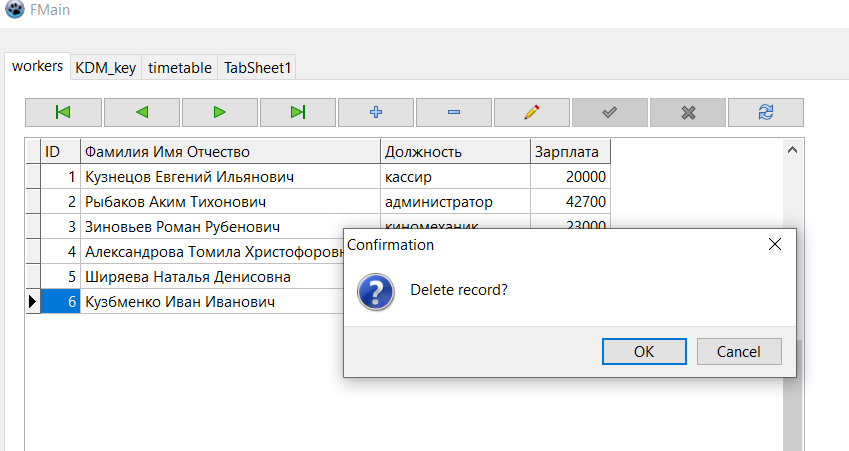


Рисунок 15. Удаление записи о сотруднике

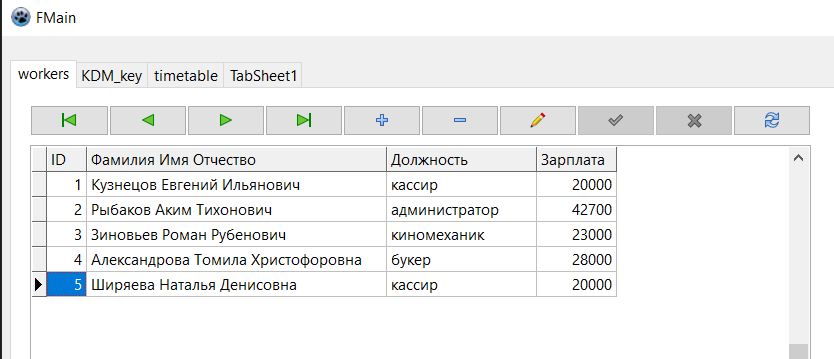


Рисунок 16. Таблица с удаленной записью

Остальные таблицы на вкладках так же успешно проходят тест.

## **Описание применения средств отладки**

В ходе написания курсового проекта при попытке запустить программу были получены следующие ошибки (рис 17)

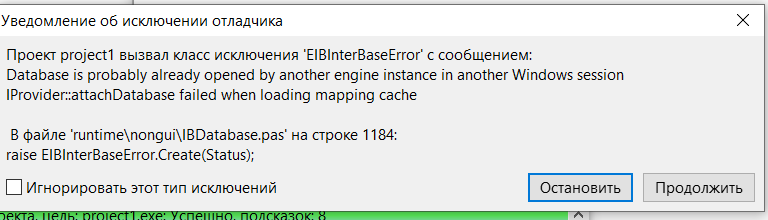


Рисунок 17. Уведомление об исключении

После определенных манипуляций, исключения были успешно ликвидированы.

## **Анализ оптимальности использования памяти и быстродействия**

Удаление мертвого кода является одним из методов оптимизации. Для быстродействия программы было решено воспользоваться данным методом. Были удалены следующие части кода:

**Листинг 2.10.1. Мертвый код в модуле umain**

procedure TFMain.movieChange(Sender: TObject);

begin

end;

**Листинг 2.10.1. Мертвый код в модуле udm**

ibqKDM.Open;

KMDIndexes:=TstringList.Create;

LoadTable('KDM\_key',nil,FMain.cmbkdm);

ibq.Close;

if not(ibq.Transaction.Active) then ibq.Transaction.StartTransaction;

ibq.sql.Text:='select \* from '+tableName;

ibq.Open;

while not(ibq.EOF) do

begin

//cmb.Items.Add(ibq.FieldByName('NAME').AsString);

ibq.Next;

end;

Размер файла project1.exe достаточно большой. Было решено оптимизировать компиляцию, тем самым уменьшить размер файла. Размер файла до оптимизации показан ниже



Рисунок 17. Размер файла до оптимизации

Для этого было необходимо отключить отладочную информацию и включить «вырезать символы из исполняемого файла».

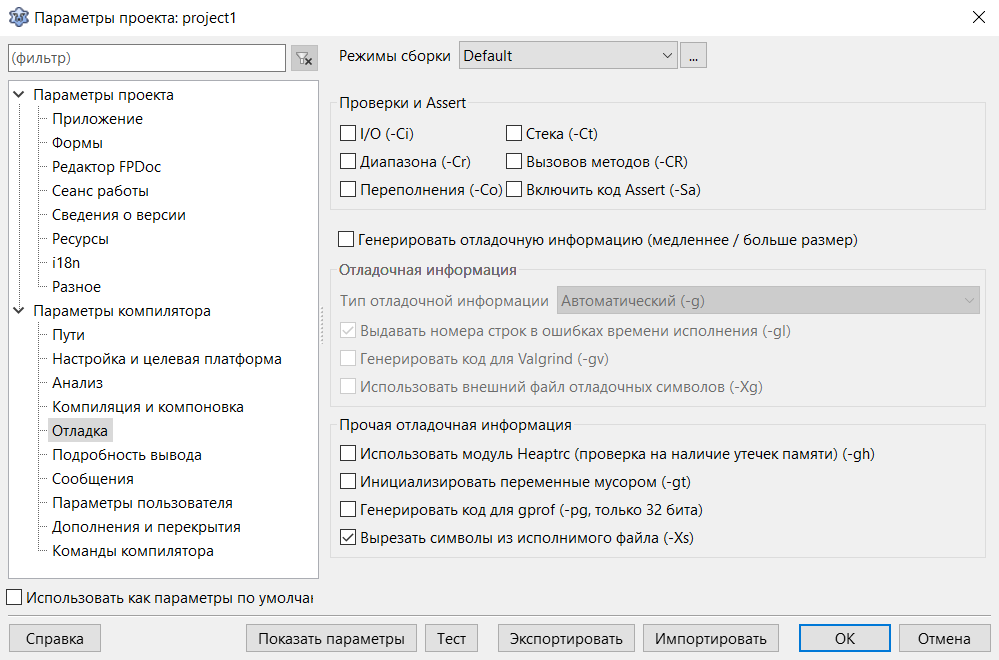


Рисунок 18. Изменение параметров отладки

Также нужно включить умную компоновку в «Стиль модуля» и в «Компоновка»

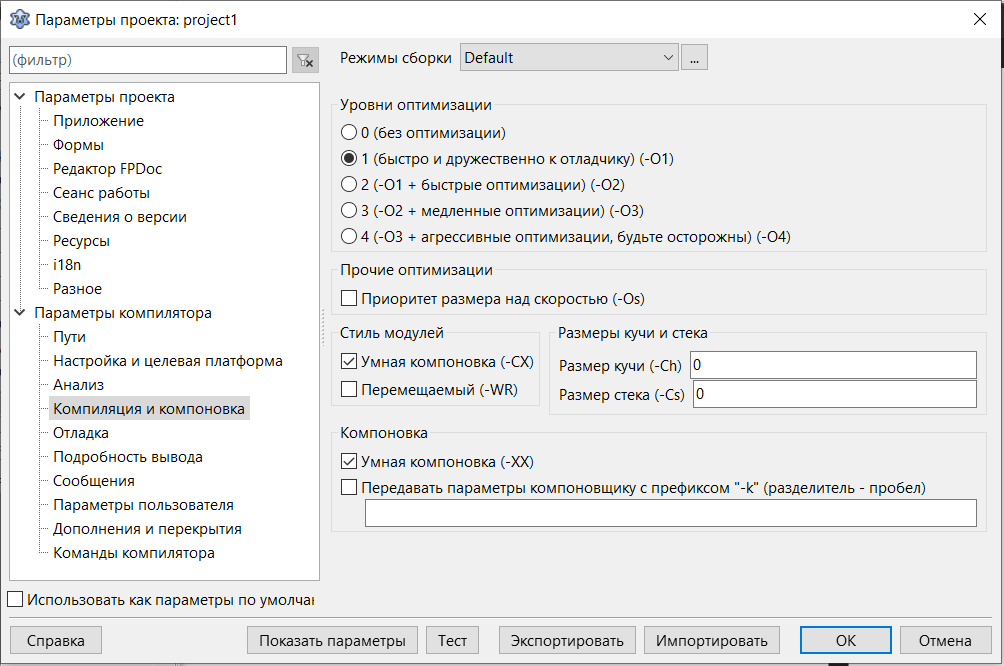


Рисунок 19. Параметры компиляции и компоновки

Для уменьшения памяти можно убрать стандартный значок.

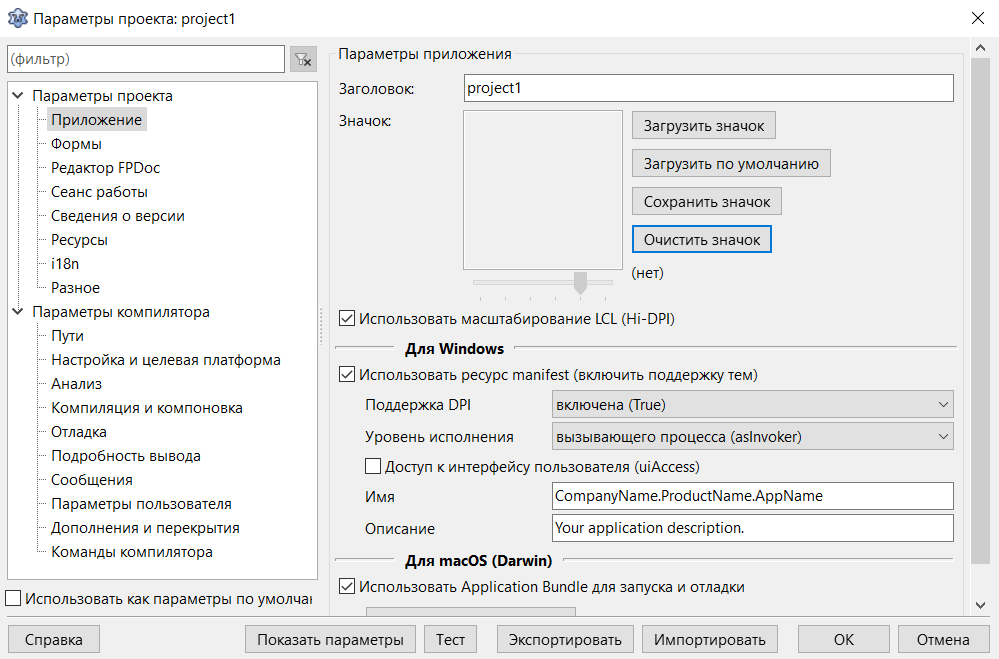


Рисунок 20. Очищение стандартного значка

После проведенной оптимизации компиляции можно заметить существенную разницу в размере файла. До оптимизации размер exe-файла составлял 46027 КБ, а после 5761 КБ



Рисунок 21. Размер файла после оптимизации

# **Глава 3. Эксплуатационная часть**

## **3.1. Руководство оператора**

### **3.1.2. Назначение программы**

Разработанное приложение позволяет осуществлять различный учет данных на предприятии быстрого питания. При этом пользователю необязательно обладать умением работы с базами данных. Программа работает с таблицами Firebird – записывает все изменения, внесенные пользователем в таблицу.

### **3.1.3. Условия выполнения программы**

Минимальные требования:

Операционная система: Windows: 7 x 32bit;

### **3.1.4. Выполнение программы**

Перед началом запуска оператор должен записать в файл DataBase.ini путь БД для работы. Если путь указан правильно, то запускается приложение project1

После запуска приложения открывается окно, содержащее четыре вкладки «workers» - работники, «KDM\_key» - KDM ключи, «timetable» - Расписание и «TabSheet1» - список фильмов

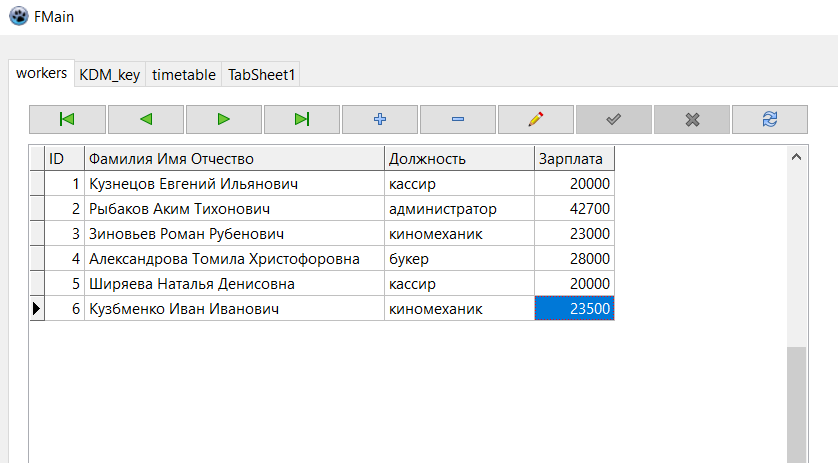


Рисунок 22. Вид приложения при открытии. Вкладка «работники»

При нажатии на одну из предложенных вкладок, открывается соответствующее окно.

Вкладка «KDM\_key» откроет таблицу «KDM ключи». (Рис. 23)

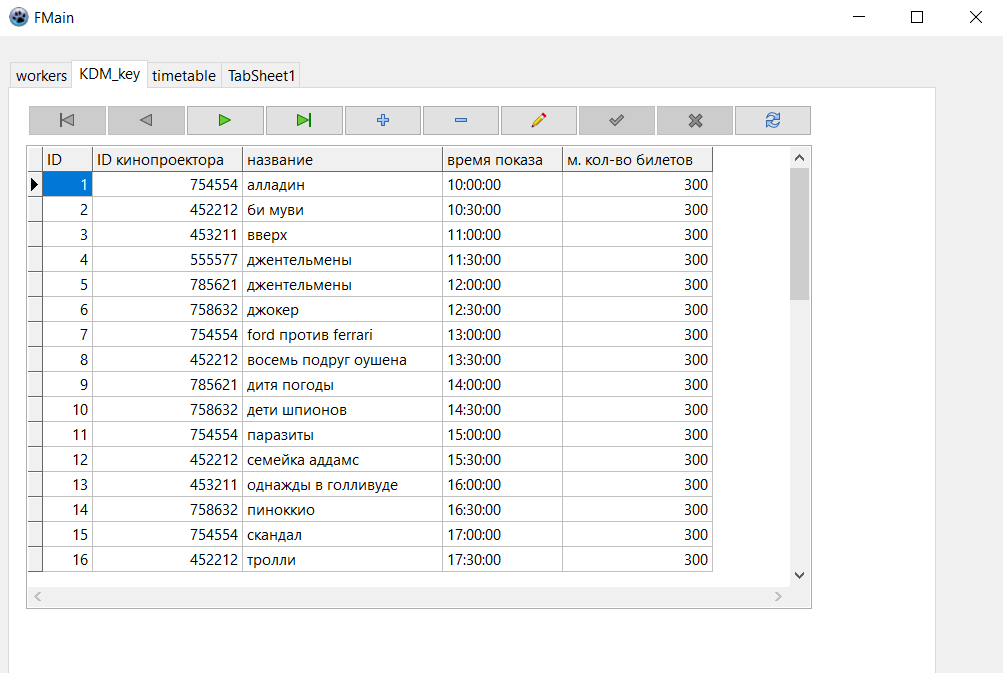


Рисунок 23. Таблица «KDM ключи»

Вкладка «timetable» откроет таблицу «расписание». (Рис. 24)

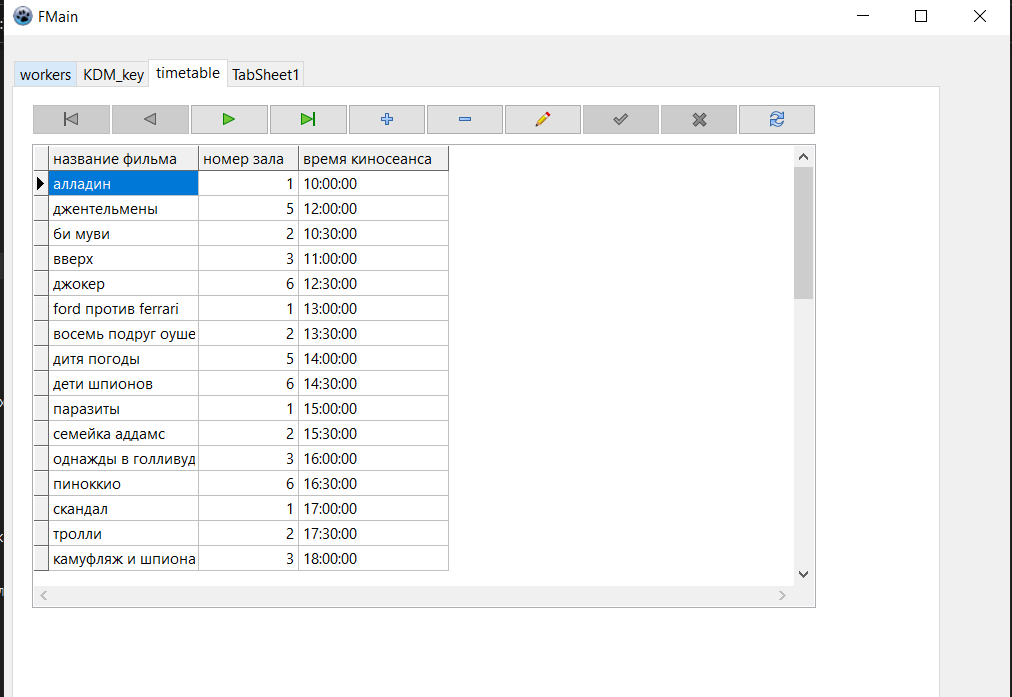


Рисунок 24. Таблица «расписание»

Вкладка «TabSheet1» откроет таблицу «фильмы». (Рис. 25)

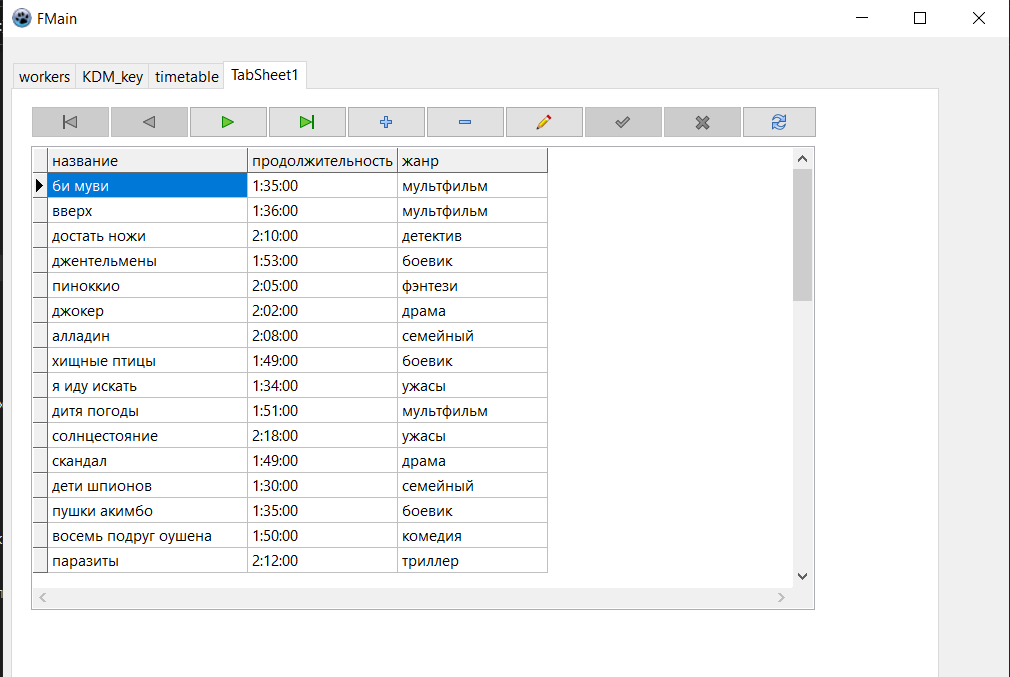


Рисунок 25. Таблица «фильмы»

# **Заключение**

В результате выполнения курсового проекта было разработано приложение «Кинотеатр».

В ходе работы были проанализированы существующие разработки посвященные данному направлению и изучены следующие темы: «Работа с базами данных FireBird» и «Разработка приложения с базами данных».

В дальнейшем планируется разработать сортировку по жанрам, возрастному цензу, добавить выпадающие списки с вариантом выбора интересующего зала и просмотра его расписание на день, добавить окно учета продажи билетов.

# **Список литературы и интернет-источников**

1. Работа в программах, программирование и многое другое...

https://www.youtube.com/channel/UCkLD\_cf5JC58UivY5Ph4Kqg

1. Национальный Открытый Университет "ИНТУИТ". Вячеслав Ачкасов. Лекция Введение в Lazarus

<http://msk.edu.ua/ivk/Informatika/Uch_posobiya/Programmirovanie/Lazarus/1_vvedenie_v_Lazarus.htm>

1. Официальный сайт ООО «Киноплан»

<https://kinoplan.ru/>

1. П1-17. МДК.01.02 Прикладное программирование. материалы лекций

<https://github.com/prog-kkkmt/p1-17/tree/master>

1. Официальный сайт Lazarus

https://www.lazarus-ide.org/

1. Официальный сайт FireBird:

<https://www.firebirdsql.org/en/server-packages/>

# **Приложения**

**Модуль umain**

unit umain;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, db, Forms, Controls, Graphics, Dialogs, DBGrids, DBCtrls,

Calendar, EditBtn, StdCtrls, DBExtCtrls, ExtDlgs, ComCtrls, PopupNotifier,

ExtCtrls, Buttons, Menus, IBTable, VirtualTrees, DBDateTimePicker,

DateTimePicker, RTTIGrids, RTTICtrls, Types;

type

{ TFMain }

TFMain = class(TForm)

DBGrid1: TDBGrid;

DBGrid2: TDBGrid;

DBGrid3: TDBGrid;

DBGrid4: TDBGrid;

DBNavigator1: TDBNavigator;

DBNavigator2: TDBNavigator;

DBNavigator3: TDBNavigator;

DBNavigator4: TDBNavigator;

movie: TPageControl;

TabSheet1: TTabSheet;

workers: TTabSheet;

KDM\_key: TTabSheet;

timetable: TTabSheet;

procedure cmbkdmChange(Sender: TObject);

procedure DBComboBox1Change(Sender: TObject);

procedure DBNavigator1Click(Sender: TObject; Button: TDBNavButtonType);

procedure sdfsClick(Sender: TObject);

// procedure movieChange(Sender: TObject);

procedure KDM\_keyContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

//procedure movieChange(Sender: TObject);

procedure TILabel1Click(Sender: TObject);

procedure timetableContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

procedure TIProgressBar1DragDrop(Sender, Source: TObject; X, Y: Integer);

procedure workersContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

private

public

end;

var

FMain: TFMain;

implementation

{$R \*.lfm}

{ TFMain }

**Модуль udm**

unit udm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, db, IBDatabase, IBTable, IBCustomDataSet, IBQuery, StdCtrls;

type

{ Tdm }

Tdm = class(TDataModule)

DataSource1: TDataSource;

dtsfilm: TDataSource;

dtsTimetable: TDataSource;

dtsWorkers: TDataSource;

ibdb: TIBDatabase;

ibfilmDURATION: TTimeField;

ibfilmGENRE: TIBStringField;

ibfilmNAME: TIBStringField;

ibkdm: TIBTable;

ibkdmAMOUNT: TIBIntegerField;

ibkdmID: TIBIntegerField;

ibkdmID\_FILM\_PROJECTOR: TIBIntegerField;

ibkdmNAME: TIBStringField;

ibkdmTIME: TTimeField;

ibqKDMID: TIBIntegerField;

ibqKDMID\_FILM\_PROJECTOR: TIBIntegerField;

ibqKDMNAMBER: TIBIntegerField;

ibqKDMNAME: TIBStringField;

ibqKDMSEAT: TIBIntegerField;

ibfilm: TIBTable;

ibTimetable: TIBTable;

IBTable1AMOUNT: TIBIntegerField;

IBTable1ID: TIBIntegerField;

IBTable1ID\_FILM\_PROJECTOR: TIBIntegerField;

IBTable1NAME: TIBStringField;

IBTable1TIME: TTimeField;

ibTimetableNAMBER: TIBIntegerField;

ibTimetableNAME: TIBStringField;

ibTimetableSESSIONTIME: TTimeField;

ibWorkers: TIBTable;

ibtr: TIBTransaction;

ibWorkersID: TIBIntegerField;

ibWorkersNAME: TIBStringField;

ibWorkersPAY: TIBBCDField;

ibWorkersPOST: TIBStringField;

procedure DataModuleCreate(Sender: TObject);

procedure DataSource1DataChange(Sender: TObject; Field: TField);

procedure dtsfilmDataChange(Sender: TObject; Field: TField);

procedure dtsKDMDataChange(Sender: TObject; Field: TField);

procedure dtsTimetableDataChange(Sender: TObject; Field: TField);

procedure dtsWorkersDataChange(Sender: TObject; Field: TField);

private

public

skladIndexes:TstringList;

procedure LoadTable(tableName:string;strl:TstringList;cmd:Tcombobox);

end;

var

dm: Tdm;

implementation

{$R \*.lfm}

{ Tdm }

procedure Tdm.DataModuleCreate(Sender: TObject);

begin

ibtr.StartTransaction;

ibWorkers.Open;

ibTimetable.Open;

//ibqKDM.Open;

ibkdm.Open;

ibfilm.Open;

// KMDIndexes:=TstringList.Create;

// LoadTable('KDM\_key',nil,FMain.cmbkdm);

end;

**Структура БД**

