**Министерство цифрового развития, связи и массовых коммуникаций РФ**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Сибирский государственный университет телекоммуникаций и информатики»**

**(СибГУТИ)**

### **КУРСОВАЯ РАБОТА**

Вариант 29

Работа кинотеатра

Выполнил студент группы ИП-115 Милевская Алёна

Проверил доцент кафедры ПМиК к.т.н Мерзлякова Е.Ю.

Новосибирск \_\_\_\_\_\_г.

Новосибирск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc154479206)

[1. АНАЛИЗ ЗАДАЧ И ПОЛЬЗОВАТЕЛЕЙ 4](#_Toc154479207)

[2. ВЫБОР РЕПРЕЗЕНТАТИВНЫХ ЗАДАЧ 5](#_Toc154479208)

[3. ЗАИМСТВОВАНИЕ 6](#_Toc154479209)

[4. ПРОТОТИП ИНТЕРФЕЙСА 8](#_Toc154479210)

[5. ФУНКЦИОНАЛ 11](#_Toc154479211)

[CWT-анализ 15](#_Toc154479212)

[GOMS-анализ 17](#_Toc154479213)

[Список используемых источников 20](#_Toc154479214)

[Листинг 21](#_Toc154479215)

# ВВЕДЕНИЕ

Целью данной курсовой работы является создание приложения «Работа кинотеатра», в котором применяются изученные навыки при прохождении курса «Визуальное программирование и человеко-машинное взаимодействие», включающих в себя разработку интерфейса, CWT и GOMS анализы, работа с базами данных, графикой, библиотеками и т.п.

Курсовая работа представляет собой приложение, позволяющее работнику кассы кинотеатра управлять списком представленных фильмов с возможностью добавления, удаления и изменения, и бронировать для посетителей интересующие их фильмы на нужное время с возможностью выбора мест. На случай, если возникнут проблемы с пониманием интерфейса предусмотрено окошко «Помощь» с описанием действий и расшифровкой сочетаний клавиш.

Мной была выбрана данная тема, так как я нередко бываю в кино и внешний вид и принцип работы подобных программ мне знаком, так как обычно я бронирую места не через кассира, а на специальном электронном стенде. Это позволило проанализировать работу данных стендов и понять, какой функционал будет наиболее понятен простым пользователям, не знакомым с электроникой.

1. АНАЛИЗ ЗАДАЧ И ПОЛЬЗОВАТЕЛЕЙ

Так как программой может воспользоваться человек любой возрастной категории, начиная со школьного возраста, для анализа были выбраны люди, которые имеют интерес к просмотру фильмов и кинотеатрам:

* Брат: возраст – 30 лет, образование – высшее, заместитель директора, навыки в выбранной сфере – нечастое бронирование фильмов в кинотеатре и выбор фильмов на сайтах, навыки владения компьютером - средние.
* Папа: возраст – 55 лет, образование – два высших, директор, навыки в выбранной сфере – часто смотрит фильмы и сериалы в интернете, навыки владения компьютером – высокие.

Пользователями программного продукта могут быть люди любых возрастов, обычно начиная со школьного, навыки владения компьютера могут варьироваться, но чаще базовые.

Возможности существующих программ: изменение списка фильмов, бронирование мест по времени.

1. ВЫБОР РЕПРЕЗЕНТАТИВНЫХ ЗАДАЧ

Определим главные задачи:

1. Добавление фильмов с возможностью внесения всех необходимых данных о нем.
2. Добавление обложки фильмов.
3. Добавления описания фильмов для примерного объяснения его смысла.
4. Удаление фильмов.
5. Редактирование уже созданных записей о фильмах.
6. Бронирование мест на определенный фильм с указанием фамилии, даты и времени.
7. Создание окошка с объяснением представленных действий.
8. ЗАИМСТВОВАНИЕ

Сайты кинотеатров обычно имеют список представленных в них фильмов, именно это послужило прототипом для создания списка с изображением и другими описаниями фильма.

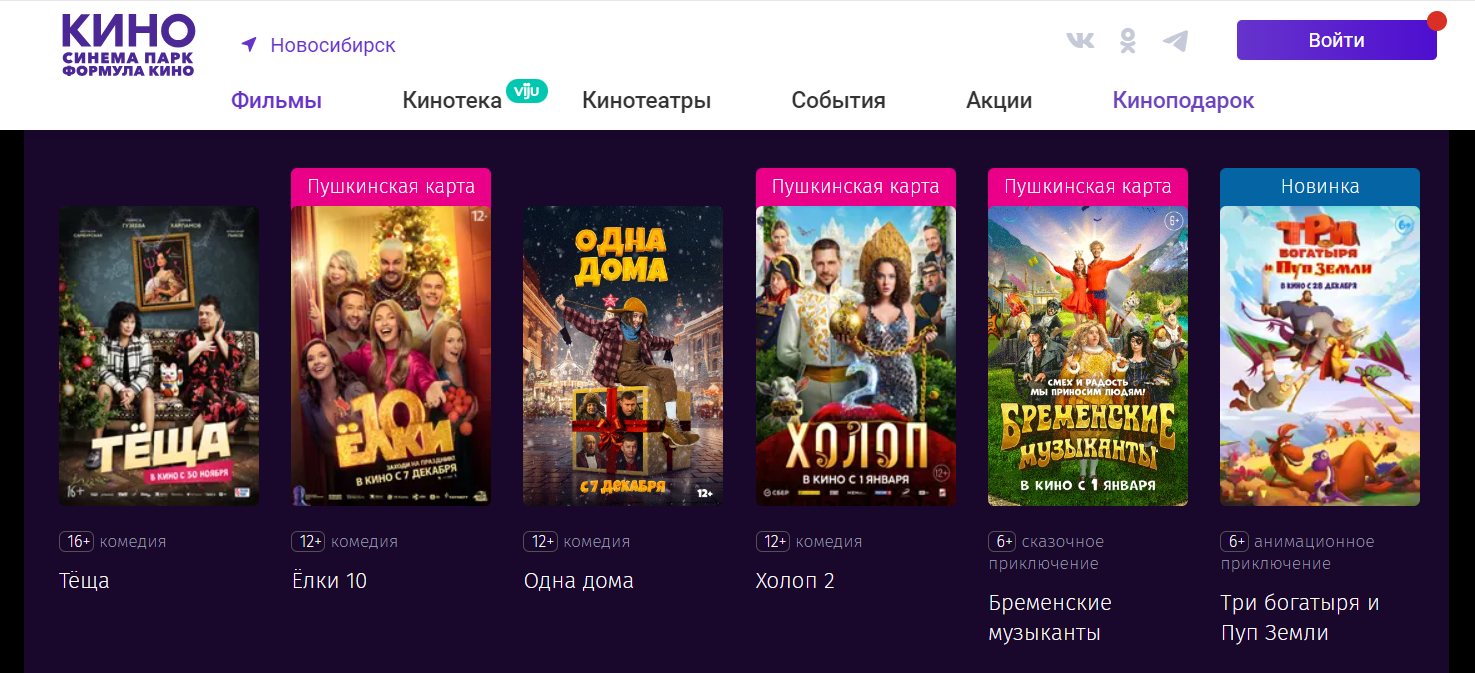


Рисунок 1. Первое окно сайта kinoteatr.ru с описанием фильмов

Также при выборе фильма появляется возможность брони места:

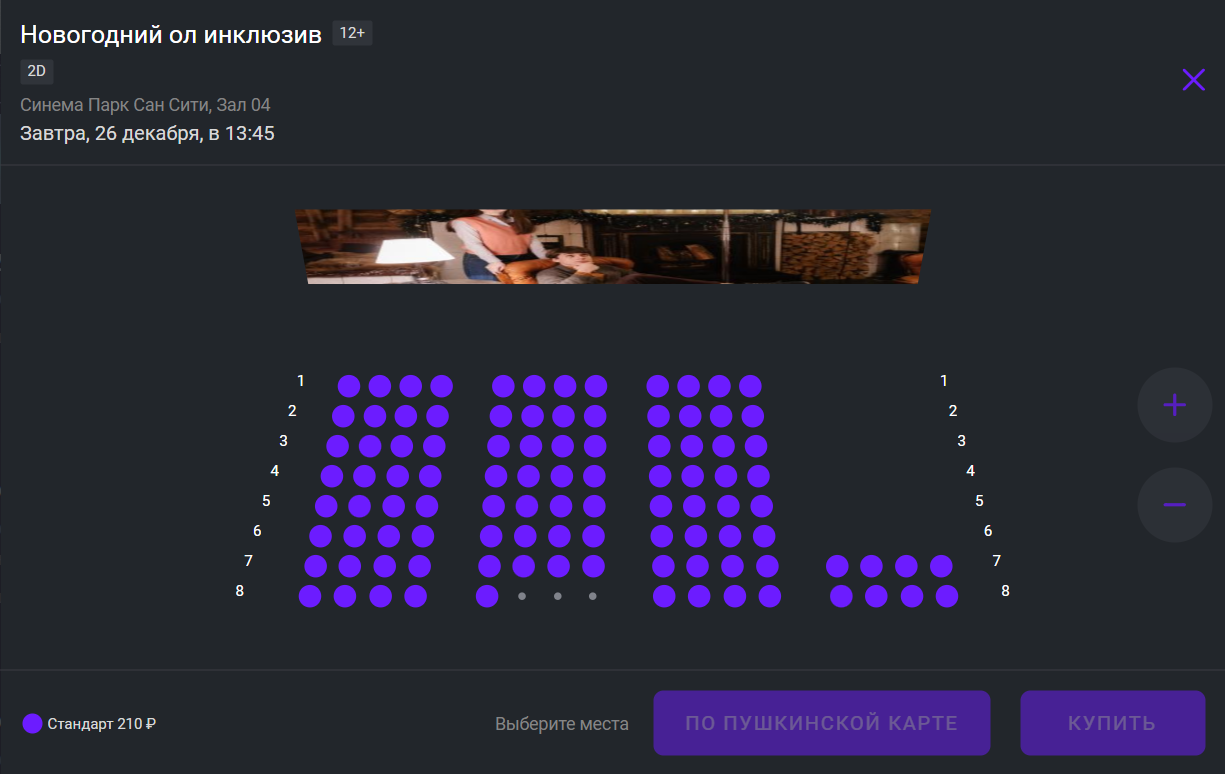


Рисунок 2. Второе окно сайта kinoteatr.ru с выбором места, даты, времени

1. ПРОТОТИП ИНТЕРФЕЙСА

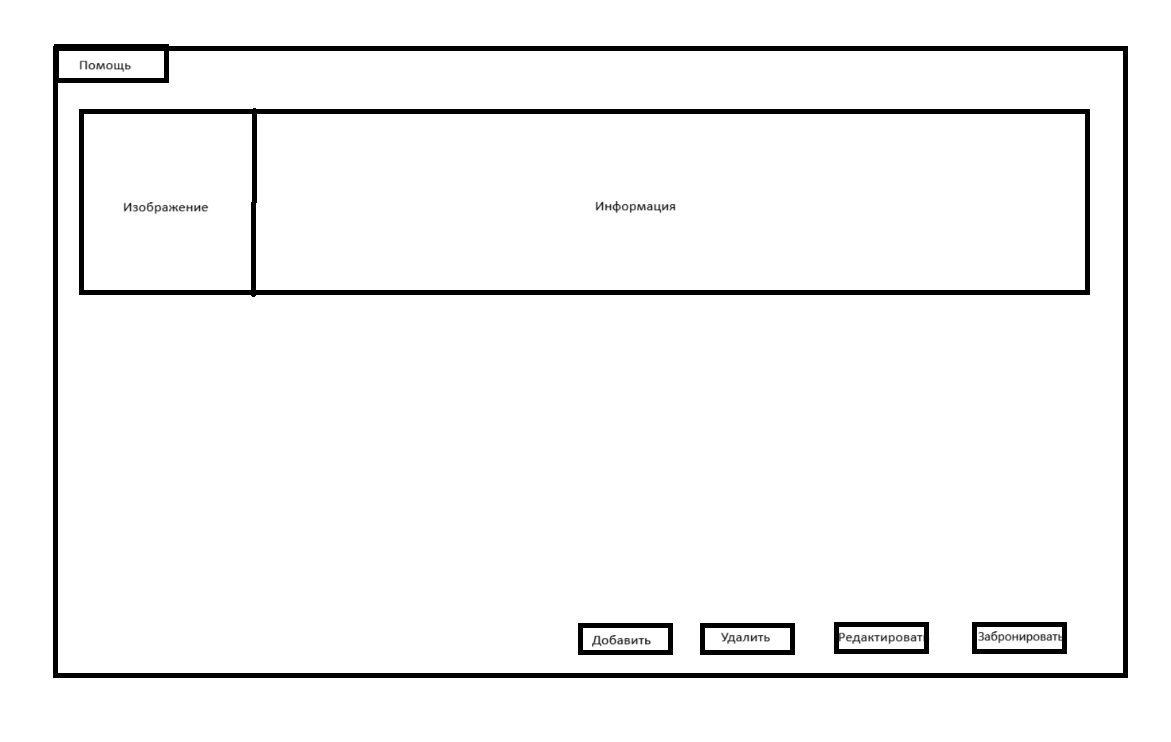


Рисунок 3. Прототип главной страницы

На главном экране будет появляться список введенных в базу ранее фильмов. Будет показываться изображение и остальная информация о фильме. В левом верхнем углу будет доступна вкладка «Помощь», где будут описана информация по каждому действию. Кнопка «Добавить» открывает новое окно, где уже будет происходить занесение данных о фильме. Кнопка «Удалить» позволяет удалить имеющийся фильм. «Редактировать» откроет новое окно, такое же как и у кнопки «Добавить», только с функцией редактирования. Кнопка «Забронировать» приведет к окошку бронирования билетов.

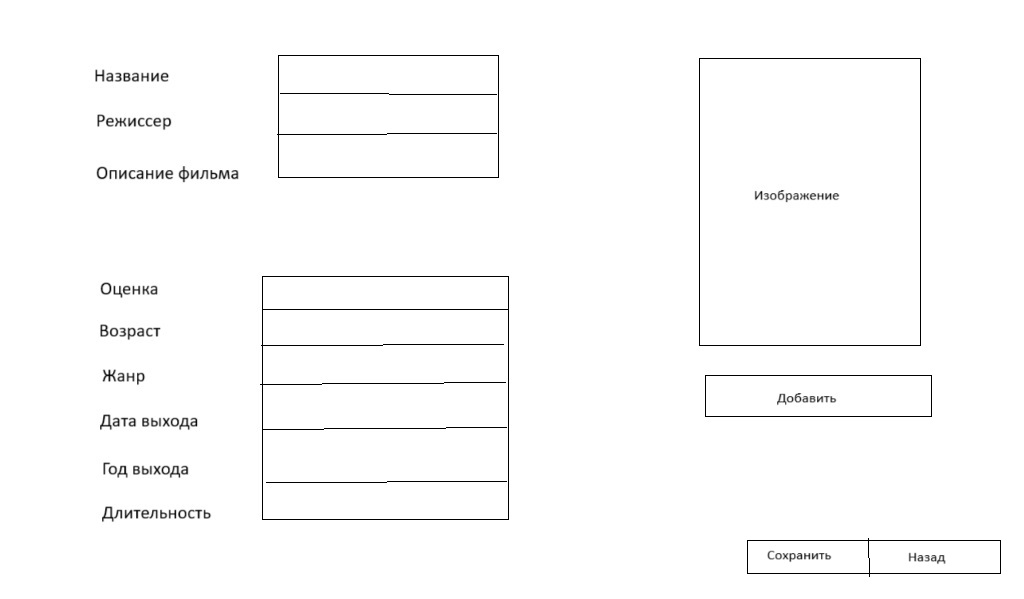


Рисунок 4. Макет добавления\редактирования фильма

В окне добавления фильма можно указать название, режиссёра, небольшое описание-аннотацию, выбрать оценку, возрастное ограничение, жанр, дату и год выхода и длительность. Также справа можно добавить фото. Кнопка «Сохранить» добавит фильм в список, а «Назад» поможет выйти.

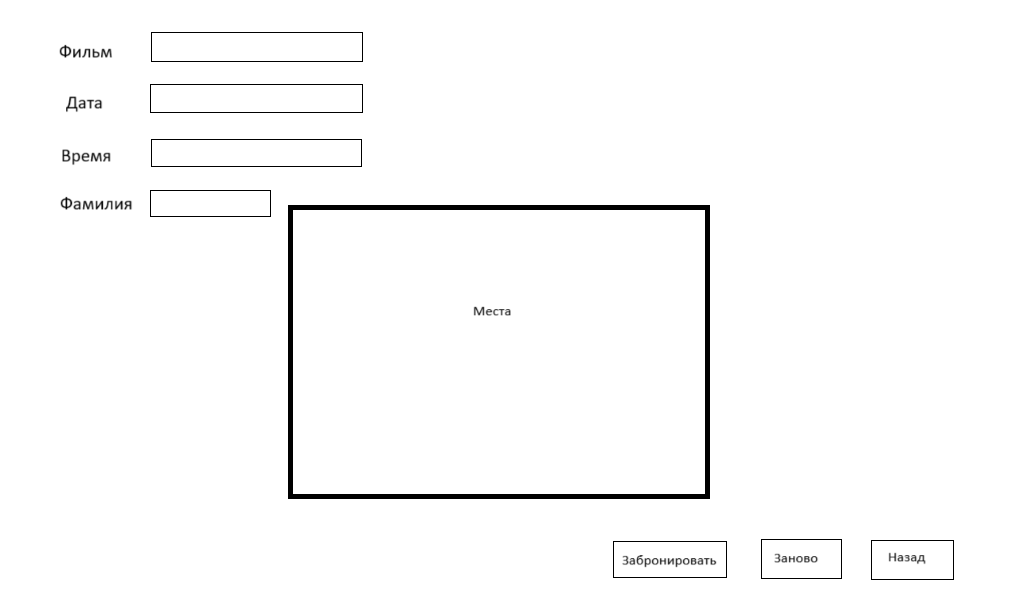


Рисунок 5. Макет окна бронирования

В окне бронирования можно выбрать название фильма из предложенных, выбрать дату и время брони, записать фамилию и выбрать место. Роль мест будут играть кнопки в виде рядов кинозала. Кнопка «Забронировать» добавит бронь в базу, «Заново» позволит сбросить значения, «Назад» позволит выйти в главное меню.

1. ФУНКЦИОНАЛ

Программа будет выполнять функционал:

1. Добавление фильмов.
2. Удаление фильмов.
3. Просмотр списка всех фильмов.
4. Редактирование фильмов.
5. Бронирование мест.
6. Сохранение брони в базу данных.

Приложение состоит из окон:

1. Главное окно со списком фильмов
2. Окно добавления
3. Окно редактирования
4. Окно бронирования
5. Окно помощи

База данных написана на языке SQLite с использованием библиотеки QSql. Создана 1 таблица базы данных.

Скриншоты работы программы

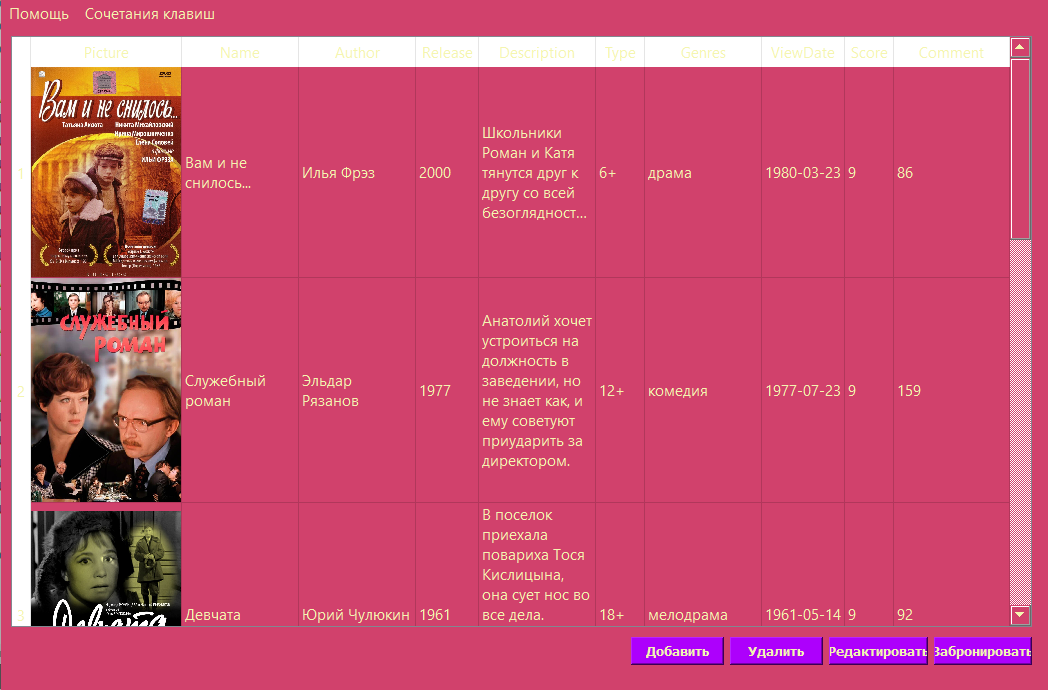


Рисунок 6. Скриншот главной страницы

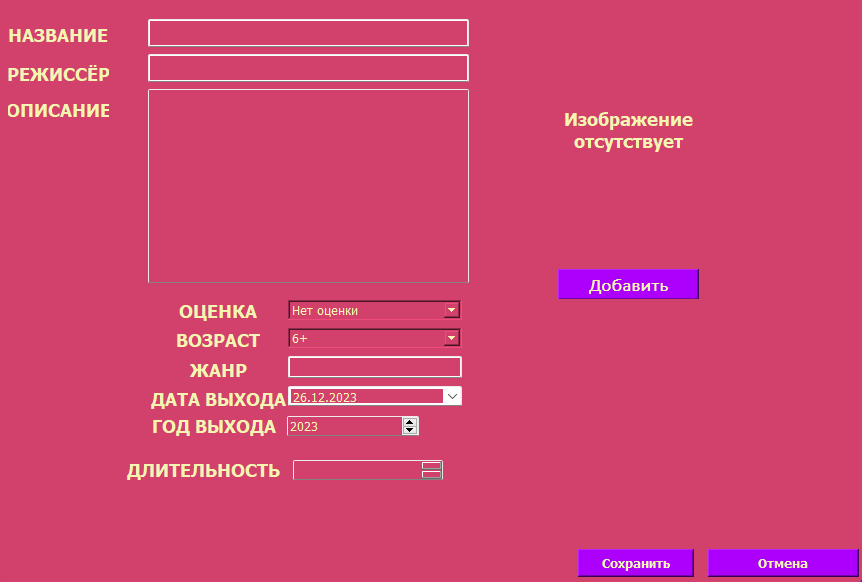


Рисунок 7. Добавление\редактирование фильма

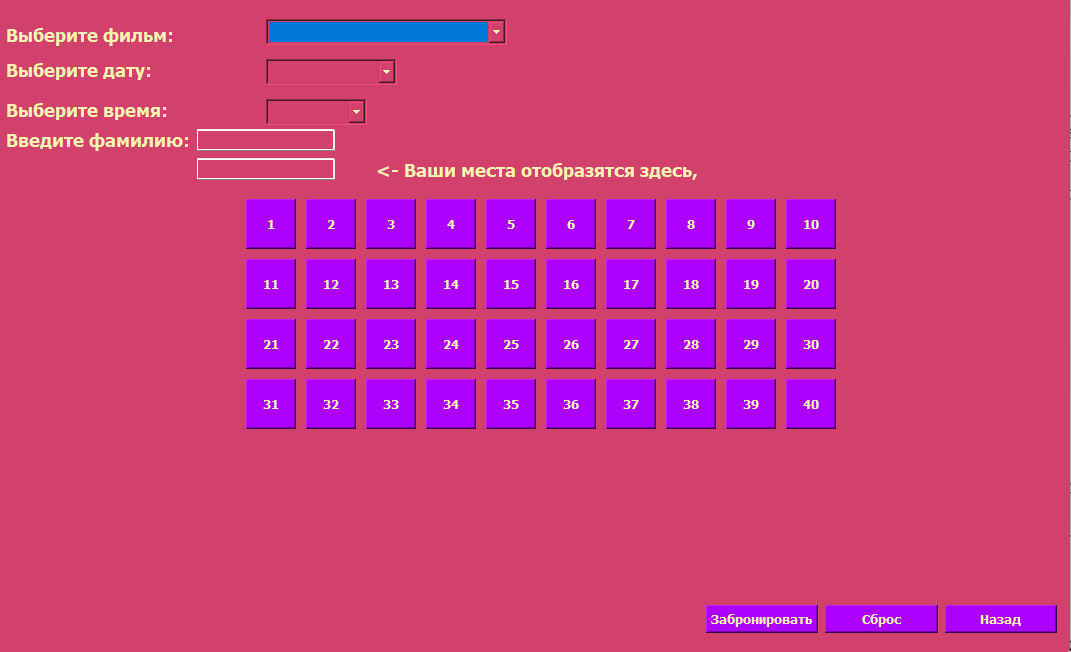


Рисунок 8. Бронирование места

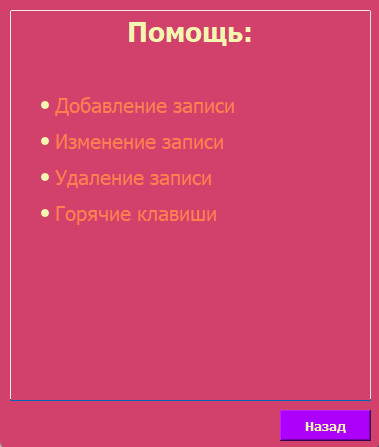


Рисунок 9. Окно помощи

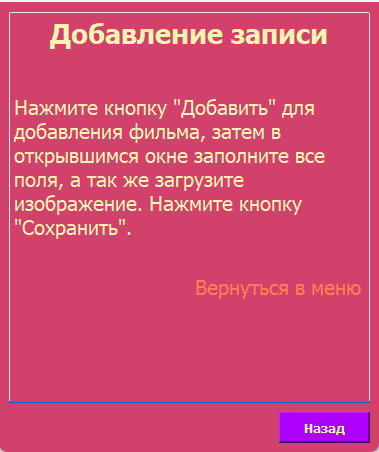


Рисунок 10. Вкладка окна помощи

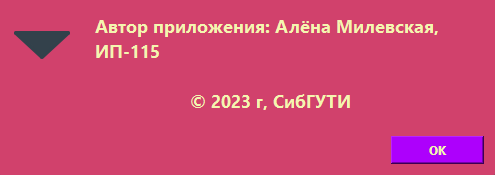


Рисунок 11. Окно с информацией о создателе

# CWT-анализ

Задача 1. Бронирование фильма

Список действий:

1. Запустить приложение
2. Нажать кнопку «Забронировать»
3. Ввести данные
4. Выбрать места
5. Нажать кнопку «Забронировать»

В 4 пункте может возникнуть проблема, что пользователь попытается ввести данные вручную, а не нажать на кнопки, так как поле выглядит идентично полю ввода. Эту проблему можно исправить, добавив поясняющую надпись или изменив поле, где отображаются выбранные места.

После нажатия кнопки «Забронировать» окно не закрывается, и если пользователь хочет продолжить выполнять действия в главном меню, ему придется самому закрыть окно бронирования, что может показаться ему равносильно потере брони. Это можно исправить добавлением функции автоматического закрытия окна при бронировании.

Задача 2. Отмена добавления

Список действий:

1. Запустить приложение
2. Нажать на кнопку «Добавить»
3. Заполнить поля
4. Добавить изображение
5. Нажать на кнопку «Отмена»

Анализ действий:

Возникает проблема в отсутствии предупреждения о сбросе данных. Пользователь может случайно нажать кнопку «Отмена» и все его данные пропадут, придется заполнять все заново.

Для решения этой проблемы необходимо добавить предупреждение о том, что данные так не сохранятся.

# GOMS-анализ

Описание анализа

Практически все интерфейсные взаимодействия можно описать следующими операциями:

K – нажатие клавиши;

B – клик кнопкой мыши;

P – наведение указателя мыши;

R – ожидание ответной реакции компьютера;

H – перенос руки с клавиатуры на мышь или наоборот

D – проведение с помощью мыши прямой линии (например, выделение или прокрутка текста);

M – мыслительная подготовка (к осуществлению одной из перечисленных операций).

Разные пользователи выполняют указанные операции за разное время. Однако, GOMS исследует работу опытного пользователя. Многочисленные исследования выявили средние значения времени операций, выполняемых опытными пользователями.

K 0.2 с

B 0.2 с

P 1.1 с

H 0.4 с

M 1.35 с

Цель 1. Изменение изображения в записи с сохранением.

Подцели:

1. Нажать кнопку «Добавить новое»
2. Выбрать в папке нужное изображение
3. Нажать кнопку «Сохранить»

Операции:

1. Нажать кнопку «Добавить новое»

MPB (1.35 + 1.1 + 0.2 = 2.65)

1. Выбрать в папке нужное изображение и кликнуть 2 раза

MPBB (1.35 + 1.1 + 0.2 + 0.2 = 2.85)

1. Нажать кнопку «Сохранить»

MPB (1.35 + 1.1 + 0.2 = 2.65)

Итог:

**2.65 + 2.85 + 2.65 = 8.15 c**

Цель 2. Открытие информации о добавлении записи.

Подцели:

1. Нажать на кнопку «Помощь»
2. Выбрать пункт «Помощь»
3. Выбрать пункт «Добавление записи»

Операции:

1. Нажать на кнопку «Помощь»

MPB (1.35 + 1.1 + 0.2 = 2.65)

1. Выбрать пункт «Помощь»

PB (1.1 + 0.2 = 1.3)

1. Выбрать пункт «Добавление записи»

PB (1.1 + 0.2 = 1.3)

Итог:

**2.65 + 1.3 + 1.3 = 5.25 с**

# Список используемых источников

1. Сайт kinoteatr.ru как прототип
2. Программа SQLiteStudio для просмотра и редактирования базы даных

# Листинг

Подключение к базе данных:

class Tickets : public QDialog

{

Q\_OBJECT

public:

QSqlDatabase mydb;

void **connClose**()

{

mydb.close();

mydb.removeDatabase(QSqlDatabase::defaultConnection);

}

bool **connOpen**()

{

mydb=QSqlDatabase::addDatabase("QSQLITE");

mydb.setDatabaseName("C:/Users/milev/Desktop/VisualProg/curs/db/kino.db");

//проверка на чтение БД

if(mydb.open()) {

qDebug("Connected...");

return true;

}

else {

qDebug("Failed to open the database");

return false;

}

}

public:

explicit Tickets(QWidget \*parent = 0);

~***Tickets***();

Добавление фильма (загрузка изображения, предупреждения)

void AddRecord::**on\_CancelButton\_clicked**()

{

ui->picture->clear();

ui->picture->setText("Изображение\nотсутствует");

ui->nameLineEdit->clear();

ui->authorLineEdit->clear();

ui->descriptionTextEdit->clear();

ui->typeComboBox->setCurrentIndex(0);

ui->genresLineEdit->clear();

ui->scoreComboBox->setCurrentIndex(0);

ui->commentTextEdit->clear();

pathImage = "";

ui->viewDateEdit->setDate(QDate::currentDate());

ui->releaseDateEdit->setDate(QDate::currentDate());

close();

}

void AddRecord::**on\_SaveButton\_clicked**()

{

QString name = ui->nameLineEdit->text();

if (name == ""){

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Название\"");

return;

}

QString author = ui->authorLineEdit->text();

if (author == ""){

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Режиссер\"");

return;

}

int releaseYear = ui->releaseDateEdit->date().year();

QString description = ui->descriptionTextEdit->toPlainText();

if (description == ""){

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Описание\"");

return;

}

QString type = ui->typeComboBox->currentText();

QString genres = ui->genresLineEdit->text();

if (genres == ""){

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Жанр\"");

return;

}

QDate viewDate = ui->viewDateEdit->date();

qint8 score = (qint8) ui->scoreComboBox->currentIndex();

QString comment = ui->commentTextEdit->toPlainText();

if (pathImage == ""){

QMessageBox::information(nullptr, "Уведомление", "Загрузите изображение");

return;

}

Редактирование фильма

void EditRecord::**on\_uploadPicture\_clicked**()

{

pathImage = QFileDialog::getOpenFileName(nullptr, "Выберите изображение", QDir::currentPath(), "\*.png \*.jpg \*.gif \*.jpeg \*.webp");

if (pathImage != "") {

// Загрузка выбранного изображения

QImage image(pathImage);

QPixmap pic = QPixmap::fromImage(image);

qDebug() << pathImage;

QPixmap inPixmap;

inPixmap.load(pathImage);

QByteArray inByteArray;

QBuffer inBuffer(&inByteArray);

inBuffer.*open*(QIODevice::WriteOnly);

inPixmap.save(&inBuffer, "PNG");

savePic = inByteArray;

ui->picture->setPixmap(pic);

ui->picture->setPixmap(pic.scaled(ui->picture->width(), ui->picture->height(), Qt::KeepAspectRatio));

}

}

void EditRecord::**catchInfo**(const int &id, const QByteArray &pic, const QString &name, const QString &author, const int &releaseYear, const QString &description, const QString &type, const QString &genres, const QDate &viewDate, const qint8 &score, const QString &comment)

{

this->id = id;

ui->nameLineEdit->setText(name);

ui->authorLineEdit->setText(author);

ui->releaseDateEdit->setDate(QDate(releaseYear, 1, 1));

ui->descriptionTextEdit->setText(description);

ui->typeComboBox->setCurrentText(type);

ui->genresLineEdit->setText(genres);

ui->viewDateEdit->setDate(viewDate);

ui->scoreComboBox->setCurrentIndex(score);

ui->commentTextEdit->setText(comment);

QPixmap img = QPixmap();

savePic = pic;

img.loadFromData(pic);

ui->picture->setPixmap(img.scaled(ui->picture->width(), ui->picture->height(), Qt::KeepAspectRatio));

}

void EditRecord::**on\_CancelButton\_clicked**()

{

this->close();

}

void EditRecord::**on\_SaveButton\_clicked**()

{

QString name = ui->nameLineEdit->text();

if (name == "") {

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Название\"");

return;

}

QString author = ui->authorLineEdit->text();

if (author == "") {

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Автор(-ы)\"");

return;

}

int releaseYear = ui->releaseDateEdit->date().year();

QString description = ui->descriptionTextEdit->toPlainText();

if (description == "") {

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Описание\"");

return;

}

QString type = ui->typeComboBox->currentText();

QString genres = ui->genresLineEdit->text();

if (genres == "") {

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Жанр(-ы)\"");

return;

}

QDate viewDate = ui->viewDateEdit->date();

qint8 score = (qint8)ui->scoreComboBox->currentIndex();

QString comment = ui->commentTextEdit->toPlainText();

if (comment == "") {

QMessageBox::information(nullptr, "Уведомление", "Заполните поле \"Комментарий\"");

return;

}

QByteArray pic = savePic;

emit sendToWidgetUpdate(id, pic, name, author, releaseYear, description, type, genres, viewDate, score, comment);

QMessageBox::information(nullptr, "Уведомление", "Запись успешно изменена!");

Добавление окна помощи

HelpInformation::**HelpInformation**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::HelpInformation)

{

ui->setupUi(this);

this->setMaximumSize(this->width(), this->height());

this->setMinimumSize(this->width(), this->height());

connect(ui->textBrowser, SIGNAL(anchorClicked(QUrl)), this, SLOT(on\_textBrowser\_anchorClicked(QUrl)));

connect(ui->backButton, SIGNAL(clicked()), ui->textBrowser, SLOT(backward()));

connect(ui->textBrowser, SIGNAL(backwardAvailable(bool)), ui->backButton, SLOT(setEnabled(bool)));

ui->textBrowser->*setSource*(QUrl::fromLocalFile(":/helpInformation/index.html"));

}

Создание таблицы с фильмами

model = new QSqlTableModel;

model->*setTable*(TABLE);

model->*setHeaderData*(0, Qt::Horizontal, "id");

ui->tableView->*setModel*(model);

ui->tableView->setColumnHidden(0, true);

ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);

ui->tableView->setSelectionMode(QAbstractItemView::SingleSelection);

updateTable();

editForm = new EditRecord();

editForm->setWindowModality(Qt::ApplicationModal);

addForm = new AddRecord();

addForm->setWindowModality(Qt::ApplicationModal);

connect(addForm, SIGNAL(sendToWidget(QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)),

this, SLOT(addRecordDataBase(QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)));

connect(this, SIGNAL(sendForEdit(int, QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)),

editForm, SLOT(catchInfo(int, QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)));

connect(editForm, SIGNAL(sendToWidgetUpdate(int, QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)),

this, SLOT(editRecordDataBase(int, QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)));

connect(ui->menu1\_add\_record, SIGNAL(triggered()), this, SLOT(slotAdd()));

connect(ui->addButton, SIGNAL(clicked()), this, SLOT(slotAdd()));

connect(ui->menu1\_edit\_record, SIGNAL(triggered()), this, SLOT(slotEdit()));

connect(ui->editButton, SIGNAL(clicked()), this, SLOT(slotEdit()));

connect(ui->menu1\_del\_record, SIGNAL(triggered()), this, SLOT(slotDel()));

connect(ui->delButton, SIGNAL(clicked()), this, SLOT(slotDel()));

connect(ui->menu1\_exit, SIGNAL(triggered()), qApp, SLOT(quit()));

connect(ui->menu2\_help, SIGNAL(triggered()), this, SLOT(slotInfo()));

connect(ui->menu2\_about, SIGNAL(triggered()), this, SLOT(slotAbout()));

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**updateTable**()

{

currRow = -1;

currID = -1;

model->*select*();

model->*setSort*(8, Qt::AscendingOrder);

if (currRow == -1) {

ui->menu1\_edit\_record->setEnabled(false);

ui->menu1\_del\_record->setEnabled(false);

ui->editButton->setEnabled(false);

ui->delButton->setEnabled(false);

} else {

ui->menu1\_edit\_record->setEnabled(true);

ui->menu1\_del\_record->setEnabled(true);

ui->editButton->setEnabled(true);

ui->delButton->setEnabled(true);

}

if (model->*rowCount*() == 0) {

ui->label->show();

ui->tableView->hide();

return;

}

ui->label->hide();

ui->tableView->show();

QPixmap pic = QPixmap();

for (int i = 0; i < model->*rowCount*(); i++) {

pic.loadFromData(model->*data*(model->*index*(i, 1)).toByteArray());

QLabel \*imageLabel = new QLabel();

imageLabel->setPixmap(pic.scaled(150, 400, Qt::KeepAspectRatio));

ui->tableView->setIndexWidget(model->*index*(i, 1), imageLabel);

}

ui->tableView->horizontalHeader()->setSectionResizeMode(1, QHeaderView::ResizeToContents);

ui->tableView->horizontalHeader()->setSectionResizeMode(2, QHeaderView::Stretch);

ui->tableView->horizontalHeader()->setSectionResizeMode(3, QHeaderView::Stretch);

ui->tableView->horizontalHeader()->setSectionResizeMode(4, QHeaderView::ResizeToContents);

ui->tableView->horizontalHeader()->setSectionResizeMode(5, QHeaderView::Stretch);

ui->tableView->horizontalHeader()->setSectionResizeMode(6, QHeaderView::ResizeToContents);

ui->tableView->horizontalHeader()->setSectionResizeMode(7, QHeaderView::Stretch);

ui->tableView->horizontalHeader()->setSectionResizeMode(8, QHeaderView::ResizeToContents);

ui->tableView->horizontalHeader()->setSectionResizeMode(9, QHeaderView::ResizeToContents);

ui->tableView->horizontalHeader()->setSectionResizeMode(10, QHeaderView::Stretch);

ui->tableView->verticalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents);

ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);

}

void MainWindow::**slotAdd**()

{

addForm->show();

}

void MainWindow::**addRecordDataBase**(const QByteArray &pic, const QString &name, const QString &author, const int &releaseYear, const QString &description, const QString &type, const QString &genres, const QDate &viewDate, const qint8 &score, const QString &comment)

{

db->insertIntoTable(pic, name, author, releaseYear, description, type, genres, viewDate, score, comment);

updateTable();

currRow = -1;

currID = -1;

}

void MainWindow::**slotEdit**()

{

if (currRow != -1) {

QString name = model->*data*(model->*index*(currRow, 2)).toString();

QString author = model->*data*(model->*index*(currRow, 3)).toString();

int releaseYear = model->*data*(model->*index*(currRow, 4)).toDate().year();

QString description = model->*data*(model->*index*(currRow, 5)).toString();

QString type = model->*data*(model->*index*(currRow, 6)).toString();

QString genres = model->*data*(model->*index*(currRow, 7)).toString();

QDate viewDate = model->*data*(model->*index*(currRow, 8)).toDate();

qint8 score = static\_cast<qint8>(model->*data*(model->*index*(currRow, 9)).toInt());

QString comment = model->*data*(model->*index*(currRow, 10)).toString();

QByteArray inByteArray = model->*data*(model->*index*(currRow, 1)).toByteArray();

emit sendForEdit(currID, inByteArray, name, author, releaseYear, description, type, genres, viewDate, score, comment);

editForm->show();

}

}

void MainWindow::**editRecordDataBase**(const int &id, const QByteArray &pic, const QString &name, const QString &author, const int &releaseYear, const QString &description, const QString &type, const QString &genres, const QDate &viewDate, const qint8 &score, const QString &comment)

{

currRow = -1;

currID = -1;

db->editInTable(id, pic, name, author, releaseYear, description, type, genres, viewDate, score, comment);

updateTable();

}

Удаление фильма

void MainWindow::**slotDel**()

{

if (currRow != -1) {

db->deleteFromDatabase(currID);

updateTable();

QMessageBox::information(nullptr, "Уведомление", "Запись успешно удалена");

currRow = -1;

currID = -1;

}

}

Расписание (пример на одном фильме)

void Tickets::**showPosts**() {

//расписание Вам и не снилось...

if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "21 декабря"){

ui->comboBox\_Time->clear();

ui->comboBox\_Time->insertItem(0,"11:30");

ui->comboBox\_Time->insertItem(1,"14:10");

ui->comboBox\_Time->insertItem(2,"18:00");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "22 декабря"){

ui->comboBox\_Time->clear(); ui->comboBox\_Time->insertItem(67, "");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "23 декабря"){

ui->comboBox\_Time->clear();

ui->comboBox\_Time->insertItem(3,"17:50");

ui->comboBox\_Time->insertItem(4,"22:00");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "24 декабря"){

ui->comboBox\_Time->clear(); ui->comboBox\_Time->insertItem(68, "");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "25 декабря"){

ui->comboBox\_Time->clear(); ui->comboBox\_Time->insertItem(69, "");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "26 декабря"){

ui->comboBox\_Time->clear();

ui->comboBox\_Time->insertItem(5,"21:00");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "27 декабря"){

ui->comboBox\_Time->clear(); ui->comboBox\_Time->insertItem(70, "");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "28 декабря"){

ui->comboBox\_Time->clear(); ui->comboBox\_Time->insertItem(71, "");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "29 декабря"){

ui->comboBox\_Time->clear();

ui->comboBox\_Time->insertItem(6,"15:35");

}

else if(ui->comboBox\_Film->currentText() == "Вам и не снилось..." and ui->comboBox\_Date->currentText() == "30 декабря"){

ui->comboBox\_Time->clear(); ui->comboBox\_Time->insertItem(72, "");

}