

**СОДЕРЖАНИЕ**

Оглавление

[Анализ предметной области 2](#_Toc58877834)

[Введение 4](#_Toc58877835)

[Реализация 8](#_Toc58877836)

[Клиентская часть 20](#_Toc58877838)

[Вывод 28](#_Toc58877839)

[Список литературы 28](#_Toc58877840)

Введение

Необходимо реализовать систему для автоматизации работы Киностудии. В системе должно быть 3 типа пользователей: Директор, Администратор, Клиент.

Система должна хранить информацию о клиентах, заказах, сотрудниках, фильмах, расписание фильмов, меню продуктов. Директор может добавлять новых сотрудников и увольнять сотрудников, а также добавлять продукты в меню и на склад.

Администратор имеет такие возможности как добавление информации о фильмах, расписание и удалять.

Клиент может покупать еду, билеты и смотреть расписание сеансов, информацию о фильмах.

Анализ предметной области

Группы пользователей разрабатываемой информационной системы (ИС)

|  |  |
| --- | --- |
| № 25 | Наименование пользователя |
| 1 | Директор |
| 2 | Администратор |
| 3 | Клиент |

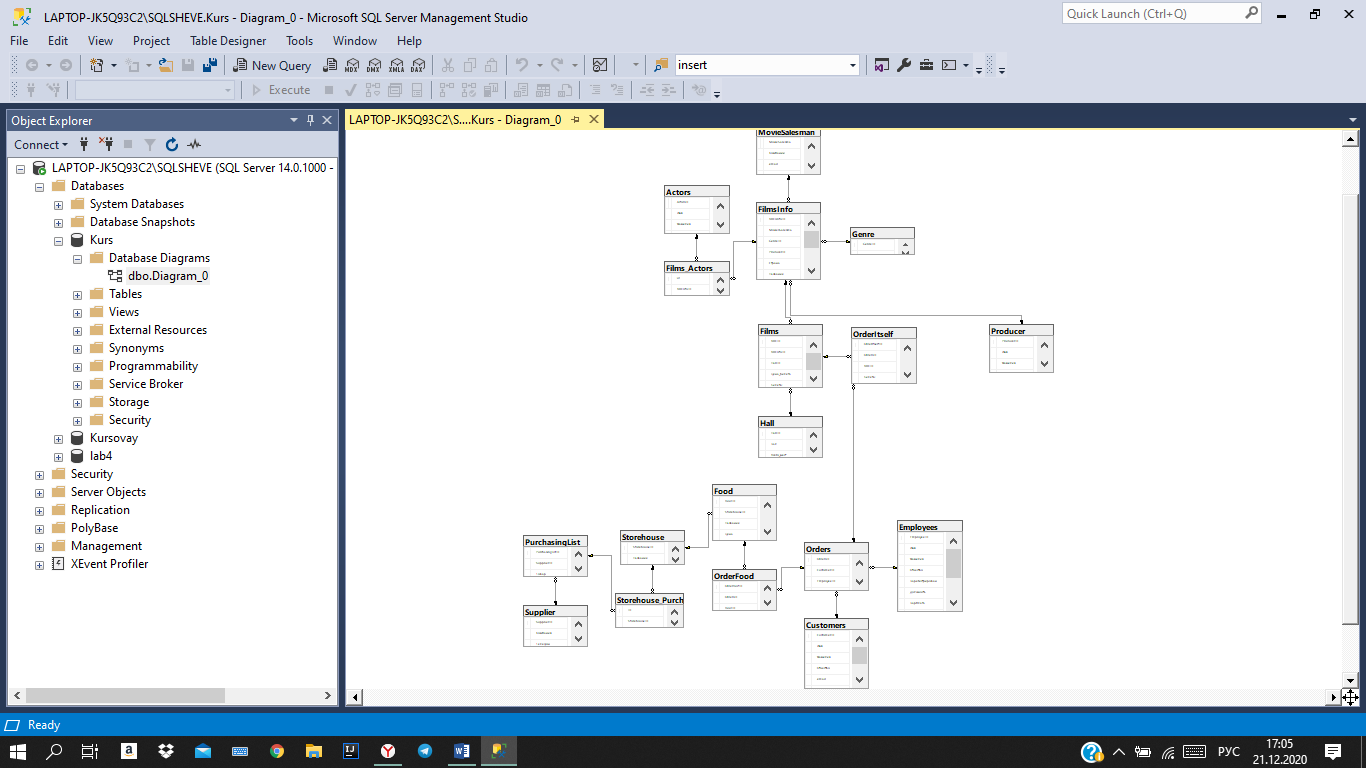
Функции групп пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № пп | Выполняемая функция | Входные данные | Выходные данные | Функции, которые должны быть реализованы в ИС |
| ***Первый пользователь(группа)*** | | | | |
| 1 | Изменять данные о сотрудниках |  |  | Добавление данных о сотрудниках и просмотр их, а также удалять их |
| 2 | Внесение данных о продуктах |  |  | Добавлять данные о продуктах в меню, склад и прайс-лист и удалять |
| ***Второй пользователь(группа)*** | | | | |
| 1 | Внесение данных о фильмах |  |  | Добавлять информацию о фильмах, продюсерах, актеров и жанров. |
| 2 | Внесение расписание |  |  | Добавлять расписание фильмов, а также добавление залов в кинотеатре |
| ***Третий пользователь(группа)*** | | | | |
| 1 | Покупка билетов |  |  | Покупать билеты на фильмы и просмотр расписание |
| 2 | Покупка еды |  |  | Покупать еду и просмотр меню |
| 3 | Просмотр и информации о фильмах |  |  | Просмотр полную информацию о фильмах |

Хранимые данные

|  |  |  |  |
| --- | --- | --- | --- |
| № пп | Хранимые данные | Пользователи, которым разрешен доступ | Ограничения по типу и значению |
| 1 | Номер заказа | Администратор, Директор | Int, not null |
| 2 | Количество человек | Администратор, Директор | Int, not null |
| 3 | ФИО гостя | Директор | Varchar, not null |
| 4 | ID Сотрудника | Директор | Int, not null |
| 5 | Название товара | Директор | Varchar, not null |
| 6 | Количество товара на складе | Директор | Int |
| 9 | Себестоимость товара | Директор | Int, not null |
| 10 | Название фильма | Администратор | Varchar, not null |
| 11 | ID фильма | Администратор, Клиент | Int, not null |
| 12 | Стоимость фильма | Клиент | Int, not null |
| 15 | Цена товара | Клиент | Int, not null |
| 16 | Счёт | Клиент | Int, not null |
| 17 | Дата заказа | Все | Date type, not null |

Исходя из анализа предметной области можем получить такую схему:



Реализация

Система реализована при помощи MS SQL Server. SQL-скрипт, создающий необходимые для обеспечения функциональности и работоспособности системы таблицы:

/\*Создание Таблиц\*/

CREATE TABLE [Employees] (

EmployeeID int NOT NULL,

Имя varchar(15) NOT NULL,

Фамилия varchar(15) NOT NULL,

Отчество varchar(15) NOT NULL,

Зарегистрирован date NOT NULL,

Должность varchar(15) NOT NULL,

Зарплата money NOT NULL,

Телефон nvarchar(20) NOT NULL,

Адрес varchar(15) NOT NULL,

Логин varchar(20) NOT NULL,

Пароль varchar(20) NOT NULL,

CONSTRAINT [PK\_EMPLOYEES] PRIMARY KEY CLUSTERED

(

[EmployeeID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Customers] (

CustomerID int NOT NULL,

Имя varchar(15) NOT NULL,

Фамилия varchar(15) NOT NULL,

Отчество varchar(15) NOT NULL,

email varchar(30) NOT NULL,

Телефон varchar(15) NOT NULL,

Логин varchar(15) NOT NULL,

Пароль varchar(15) NOT NULL,

CONSTRAINT [PK\_CUSTOMERS] PRIMARY KEY CLUSTERED

(

[CustomerID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Orders] (

OrdersID int NOT NULL,

EmployeeID int NOT NULL,

CustomerID int NOT NULL,

Дата заказа date NOT NULL,

Сумма заказа int NOT NULL,

CONSTRAINT [PK\_ORDERS] PRIMARY KEY CLUSTERED

(

[OrdersID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [OrderItself] (

OrderltselfID int NOT NULL,

OrdersID int NOT NULL,

FilmID int NOT NULL,

Ряд varchar(10) NOT NULL,

Место varchar(10) NOT NULL,

CONSTRAINT [PK\_ORDERITSELF] PRIMARY KEY CLUSTERED

(

[OrderltselfID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Films] (

FilmID int NOT NULL,

FilmsIfoID int NOT NULL,

HallID int NOT NULL,

Цена\_билета int NOT NULL,

Билеты int NOT NULL,

Дата date NOT NULL,

Время time NOT NULL,

CONSTRAINT [PK\_FILMS] PRIMARY KEY CLUSTERED

(

[FilmID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [FilmsInfo] (

FilmsInfoID int NOT NULL,

MoviesSalesmanID int NOT NULL,

GenreID int NOT NULL,

ProducerID int NOT NULL,

Страна varchar(20) NOT NULL,

Название varchar(25) NOT NULL,

Дата выпуска date NOT NULL,

Длительность time NOT NULL,

Оценка int NOT NULL,

CONSTRAINT [PK\_FILMSINFO] PRIMARY KEY CLUSTERED

(

[FilmsInfoID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [MovieSalesman] (

MovieSalesmanID int NOT NULL,

Компании varchar(30) NOT NULL,

email varchar(30) NOT NULL,

Телефон varchar(30) NOT NULL,

Адрес varchar(30) NOT NULL,

CONSTRAINT [PK\_MOVIESALESMAN] PRIMARY KEY CLUSTERED

(

[MovieSalesmanID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Hall] (

HallID int NOT NULL,

Зала varchar(15) NOT NULL,

Всего\_мест int NOT NULL,

Всего\_рядов int NOT NULL,

CONSTRAINT [PK\_HALL] PRIMARY KEY CLUSTERED

(

[HallID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Genre] (

GenreID int NOT NULL,

Жанр varchar(50) NOT NULL,

CONSTRAINT [PK\_GENRE] PRIMARY KEY CLUSTERED

(

[GenreID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Producer] (

ProducerID int NOT NULL,

Имя varchar(20) NOT NULL,

Фамилия varchar(20) NOT NULL,

Дата\_рождения date NOT NULL,

Страна varchar(20) NOT NULL,

CONSTRAINT [PK\_PRODUCER] PRIMARY KEY CLUSTERED

(

[ProducerID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Actors] (

ActorsID int NOT NULL,

Имя varchar(30) NOT NULL,

Фаимиля varchar(30) NOT NULL,

Страна varchar(30) NOT NULL,

Дата\_рождения date NOT NULL,

CONSTRAINT [PK\_ACTORS] PRIMARY KEY CLUSTERED

(

[ActorsID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Storehouse] (

StorehouseID int NOT NULL,

Название varchar(25) NOT NULL,

Количество int NOT NULL,

CONSTRAINT [PK\_STOREHOUSE] PRIMARY KEY CLUSTERED

(

[StorehouseID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Supplier] (

SupplierID int NOT NULL,

Компания varchar(25) NOT NULL,

Телефон int NOT NULL,

Адрес varchar(30) NOT NULL,

CONSTRAINT [PK\_SUPPLIER] PRIMARY KEY CLUSTERED

(

[SupplierID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Films\_Actors] (

id int NOT NULL,

FilmsIfoID int NOT NULL,

ActorsID int NOT NULL,

CONSTRAINT [PK\_FILMS\_ACTORS] PRIMARY KEY CLUSTERED

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [PurchasingList] (

PurchasingListID int NOT NULL,

SupplierID int NOT NULL,

Товар varchar(25) NOT NULL,

Количество int NOT NULL,

CONSTRAINT [PK\_PURCHASINGLIST] PRIMARY KEY CLUSTERED

(

[PurchasingListID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Storehouse\_PurchasingList] (

ID int NOT NULL,

StorehouseID int NOT NULL,

PurchasingListID int NOT NULL,

CONSTRAINT [PK\_STOREHOUSE\_PURCHASINGLIST] PRIMARY KEY CLUSTERED

(

[ID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [OrderFood] (

OrderFootID int NOT NULL,

OrdersID int NOT NULL,

FoodID int NOT NULL,

Количество int NOT NULL,

CONSTRAINT [PK\_ORDERFOOD] PRIMARY KEY CLUSTERED

(

[OrderFootID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [Food] (

FoodID int NOT NULL,

StorehouseID int NOT NULL,

Название varchar(35) NOT NULL,

Цена money NOT NULL,

Себестоимость money NOT NULL,

Количество int NOT NULL,

CONSTRAINT [PK\_FOOD] PRIMARY KEY CLUSTERED

(

[FoodID] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

ALTER TABLE [Orders] WITH CHECK ADD CONSTRAINT [Orders\_fk0] FOREIGN KEY ([EmployeeID]) REFERENCES [Employees]([EmployeeID])

ON UPDATE CASCADE

GO

ALTER TABLE [Orders] CHECK CONSTRAINT [Orders\_fk0]

GO

ALTER TABLE [Orders] WITH CHECK ADD CONSTRAINT [Orders\_fk1] FOREIGN KEY ([CustomerID]) REFERENCES [Customers]([CustomerID])

ON UPDATE CASCADE

GO

ALTER TABLE [Orders] CHECK CONSTRAINT [Orders\_fk1]

GO

ALTER TABLE [OrderItself] WITH CHECK ADD CONSTRAINT [OrderItself\_fk0] FOREIGN KEY ([OrdersID]) REFERENCES [Orders]([OrdersID])

ON UPDATE CASCADE

GO

ALTER TABLE [OrderItself] CHECK CONSTRAINT [OrderItself\_fk0]

GO

ALTER TABLE [OrderItself] WITH CHECK ADD CONSTRAINT [OrderItself\_fk1] FOREIGN KEY ([FilmID]) REFERENCES [Films]([FilmID])

ON UPDATE CASCADE

GO

ALTER TABLE [OrderItself] CHECK CONSTRAINT [OrderItself\_fk1]

GO

ALTER TABLE [Films] WITH CHECK ADD CONSTRAINT [Films\_fk0] FOREIGN KEY ([FilmsIfoID]) REFERENCES [FilmsInfo]([FilmsInfoID])

ON UPDATE CASCADE

GO

ALTER TABLE [Films] CHECK CONSTRAINT [Films\_fk0]

GO

ALTER TABLE [Films] WITH CHECK ADD CONSTRAINT [Films\_fk1] FOREIGN KEY ([HallID]) REFERENCES [Hall]([HallID])

ON UPDATE CASCADE

GO

ALTER TABLE [Films] CHECK CONSTRAINT [Films\_fk1]

GO

ALTER TABLE [FilmsInfo] WITH CHECK ADD CONSTRAINT [FilmsInfo\_fk0] FOREIGN KEY ([MoviesSalesmanID]) REFERENCES [MovieSalesman]([MovieSalesmanID])

ON UPDATE CASCADE

GO

ALTER TABLE [FilmsInfo] CHECK CONSTRAINT [FilmsInfo\_fk0]

GO

ALTER TABLE [FilmsInfo] WITH CHECK ADD CONSTRAINT [FilmsInfo\_fk1] FOREIGN KEY ([GenreID]) REFERENCES [Genre]([GenreID])

ON UPDATE CASCADE

GO

ALTER TABLE [FilmsInfo] CHECK CONSTRAINT [FilmsInfo\_fk1]

GO

ALTER TABLE [FilmsInfo] WITH CHECK ADD CONSTRAINT [FilmsInfo\_fk2] FOREIGN KEY ([ProducerID]) REFERENCES [Producer]([ProducerID])

ON UPDATE CASCADE

GO

ALTER TABLE [FilmsInfo] CHECK CONSTRAINT [FilmsInfo\_fk2]

GO

ALTER TABLE [Films\_Actors] WITH CHECK ADD CONSTRAINT [Films\_Actors\_fk0] FOREIGN KEY ([FilmsIfoID]) REFERENCES [FilmsInfo]([FilmsInfoID])

ON UPDATE CASCADE

GO

ALTER TABLE [Films\_Actors] CHECK CONSTRAINT [Films\_Actors\_fk0]

GO

ALTER TABLE [Films\_Actors] WITH CHECK ADD CONSTRAINT [Films\_Actors\_fk1] FOREIGN KEY ([ActorsID]) REFERENCES [Actors]([ActorsID])

ON UPDATE CASCADE

GO

ALTER TABLE [Films\_Actors] CHECK CONSTRAINT [Films\_Actors\_fk1]

GO

ALTER TABLE [PurchasingList] WITH CHECK ADD CONSTRAINT [PurchasingList\_fk0] FOREIGN KEY ([SupplierID]) REFERENCES [Supplier]([SupplierID])

ON UPDATE CASCADE

GO

ALTER TABLE [PurchasingList] CHECK CONSTRAINT [PurchasingList\_fk0]

GO

ALTER TABLE [Storehouse\_PurchasingList] WITH CHECK ADD CONSTRAINT [Storehouse\_PurchasingList\_fk0] FOREIGN KEY ([StorehouseID]) REFERENCES [Storehouse]([StorehouseID])

ON UPDATE CASCADE

GO

ALTER TABLE [Storehouse\_PurchasingList] CHECK CONSTRAINT [Storehouse\_PurchasingList\_fk0]

GO

ALTER TABLE [Storehouse\_PurchasingList] WITH CHECK ADD CONSTRAINT [Storehouse\_PurchasingList\_fk1] FOREIGN KEY ([PurchasingListID]) REFERENCES [PurchasingList]([PurchasingListID])

ON UPDATE CASCADE

GO

ALTER TABLE [Storehouse\_PurchasingList] CHECK CONSTRAINT [Storehouse\_PurchasingList\_fk1]

GO

ALTER TABLE [OrderFood] WITH CHECK ADD CONSTRAINT [OrderFood\_fk0] FOREIGN KEY ([OrdersID]) REFERENCES [Orders]([OrdersID])

ON UPDATE CASCADE

GO

ALTER TABLE [OrderFood] CHECK CONSTRAINT [OrderFood\_fk0]

GO

ALTER TABLE [OrderFood] WITH CHECK ADD CONSTRAINT [OrderFood\_fk1] FOREIGN KEY ([FoodID]) REFERENCES [Food]([FoodID])

ON UPDATE CASCADE

GO

ALTER TABLE [OrderFood] CHECK CONSTRAINT [OrderFood\_fk1]

GO

ALTER TABLE [Food] WITH CHECK ADD CONSTRAINT [Food\_fk0] FOREIGN KEY ([StorehouseID]) REFERENCES [Storehouse]([StorehouseID])

ON UPDATE CASCADE

GO

ALTER TABLE [Food] CHECK CONSTRAINT [Food\_fk0]

GO

/\*Заполнение таблицы\*/

Insert Employees

(Имя, Фамилия, Отчество, Должность, Зарплата, Телефон, Адрес, Логин, Пароль)

Values

('Никита', 'Шеве', 'Максимович', 'Директор', '100000', '89815553646', 'Политех, Спб', 'DIREKTOR', 'ROOT'),

('Ира', 'Цыгарова', 'Николаевна', 'Администратор', '50000', '88005554545', 'Политех, Спб', 'ADMIN\_1', 'AdminRoot'),

('Дмитрий', 'Луценко', 'Юрьевич', 'Администратор', '50000', '82281488888', 'Политех, Спб', 'ADMIN\_2', 'AdminRoot'),

('Георгий', 'Разнобарский', 'Михайлович', 'Бармен', '35000', '89817468998', 'СпбГУ, Спб', 'BAR\_1', 'BarRoot'),

('Лаки', 'Инобор', '-----', 'Бармен', '35000', '89999999999', 'Epam System, Спб', 'BAR\_2', 'BarRoot');

GO

select \* from Employees

GO

insert Actors

(Имя, Фамилия, Страна, Дата\_рождения)

values

('Робет', 'Паттинсон', 'Англия', '13.05.1986'),

('Джон', 'Дэвид', 'США', '28.07.1984'),

('Кеннет', 'Брана', 'Англия', '10.12.1960'),

('Брэд', 'Питт', 'США', '18.12.1963'),

('Тиль', 'Швайгер', 'Германия', '19.12.1963'),

('Кристоф', 'Вальц', 'Австрия', '04.10.1956'),

('Том', 'Хэнкс', 'США', '09.07.1956'),

('Стэнли', 'Туччи', 'США', '11.11.1960'),

('Кэтрин', 'Зета-Джонс', 'Англия', '25.09.1969');

go

select \* from Actors

go

select FilmsInfoID, Название from FilmsInfo

go

Insert Films\_Actors

(FilmsIfoID, ActorsID)

Values

('4', '1'),

('4', '2'),

('4', '3'),

('4', '7'),

('5', '4'),

('5', '5'),

('5', '6'),

('6', '7'),

('6', '8'),

('6', '9');

go

select \* from Films\_Actors

go

insert Hall

(Зал, Всего\_мест, Всего\_рядов)

values

('Без последнего ряда', '50', '5'),

('Райское кресло', '70', '7'),

('20 минут рекламы', '44', '4'),

('Без детей', '20', '5');

go

insert Films

(FilmsIfoID, HallID, Цена\_билета, Билеты, Дата, Время)

values

('4', '3', '100', '44', '12.12.2020', '10:00'),

('4', '3', '150', '44', '12.12.2020', '12:00'),

('4', '3', '250', '44', '12.12.2020', '16:00'),

('4', '2', '300', '70', '12.12.2020', '19:00'),

('5', '2', '120', '70', '12.12.2020', '13:00'),

('5', '1', '230', '50', '12.12.2020', '13:30'),

('5', '1', '250', '50', '12.12.2020', '16:30'),

('5', '4', '200', '20', '12.12.2020', '14:00'),

('6', '4', '300', '20', '12.12.2020', '20:00'),

('6', '2', '100', '70', '12.12.2020', '10:00'),

('6', '1', '200', '50', '12.12.2020', '21:00');

go

Так же было написано клиентское приложение, использующее следующие запросы к базе данных:

--Вывести всех режесеров и фильмы, которые они сняли

select Название, Имя, Фамилия from Producer

left join FilmsInfo on FilmsInfo.ProducerID = Producer.ProducerID

order by Название

go

--Вывести фильмы одного жанра и подсчитывающий общее число фильмов данного жанра

select Жанр, Название, count(Название) from FilmsInfo

inner join Genre on Genre.GenreID = FilmsInfo.GenreID

group by Название, Жанр

go

--Вывести сеансы в больших залах

select Название, Цена\_билета, Дата, Время from Films

inner join FilmsInfo on FilmsInfo.FilmsInfoID = Films.FilmsIfoID

where exists(select \* from Hall where Hall.HallID = Films.HallID and Hall.Всего\_мест > '40')

go

--Вывести всех актеров и режисеров из США

select Имя, Фамилия, Дата\_рождения, Страна from Producer

where Producer.Страна = 'США'

Union

(select Имя, Фамилия, Дата\_рождения, Страна from Actors

where Actors.Страна = 'США')

go

--Вывести того чего нет на скалде, но есть в прай-листе

select Товар from PurchasingList

except

select Название from Storehouse

go

--Сотрудники которые являются клиентами

select Имя, Фамилия, Отчество from Employees

intersect

select Имя, Фамилия, Отчество from Customers

go

--Фильм на который купили билетов

select distinct Название from FilmsInfo

join Films on Films.FilmsIfoID = FilmsInfo.FilmsInfoID

where exists(select \* from OrderItself where OrderItself.FilmID = Films.FilmID)

go

--Клиент который не покупал билеты на фильм

select Имя, Фамилия, Отчество from Customers

where Customers.CustomerID not in

(select Orders.CustomerID from Orders)

go

--Поставщики которые представлены в прайс-листе

select \* from Supplier

where SupplierID in

(select SupplierID from PurchasingList)

go

--Вывести ID жанров про которых нет фильмов

select GenreID from Genre

except

select GenreID from FilmsInfo

go

**Хранимые Процедуры**

--ХРАНИМЫЕ ПРОЦЕДУРЫ

--Добавление сотрудника

alter Proc insertEmployees

@Name varchar(40),

@Surname varchar(40),

@Otch varchar(40),

@Date date,

@Dol varchar(40),

@Zarplata money,

@Tel varchar(40),

@Adres varchar(40),

@Log varchar(40),

@Pass varchar(40)

as

begin

insert into Employees

(Имя, Фамилия, Отчество, Зарегистрирован, Должность, Зарплата, Телефон, Адрес, Логин, Пароль)

values

(@Name, @Surname, @Otch, @Date, @Dol, @Zarplata, @Tel, @Adres, @Log, @Pass)

end

--Удаление продукта

alter Proc deleteStorehouse @id int as

begin

delete from Storehouse where StorehouseID = @id

delete from Food where StorehouseID = @id

delete from Storehouse\_PurchasingList where StorehouseID = @id

end

--Выбор фильма

alter proc select\_Films @ID int as

select FilmsInfo.Название, FilmsInfo.Длительность, Films.Время, Films.Дата, Films.Цена\_билета, Hall.Зал from Films

inner join FilmsInfo on FilmsInfo.FilmsInfoID = Films.FilmsIfoID

inner join Hall on Hall.HallID = Films.HallID

where Films.FilmID = @ID

go

--ТРИГГЕР

--На запрет добавления данных в таблицу жанр

create trigger stoping\_Insert\_Genre on Genre for insert

as begin

if((select GenreID from inserted) > 9) rollback

end

alter proc checkPassword

@Dol varchar(40),

@Password varchar(40) as

select \* from Employees

where Employees.Должность = @Dol and Employees.Пароль = @Password

go

alter proc insertCustomers

@Name varchar(15),

@Surname varchar(15),

@Otche varchar(15),

@Email varchar(15),

@Tel varchar(15),

@Login varchar(15),

@Password varchar(15)

as begin

insert into Customers

(Имя, Фамилия, Отчество, email, Телефон, Логин, Пароль)

values

(@Name, @Surname, @Otche, @Email, @Tel, @Login, @Password)

end

alter proc insertActor

@Name varchar(30),

@Surname varchar(30),

@From varchar(30),

@Date varchar(11)

as begin

insert into Actors

(Имя, Фамилия, Страна, Дата\_рождения)

values

(@Name, @Surname, @From, @Date)

end

alter proc insertHall

@Name varchar(40),

@Mest int,

@Rad int

as begin

insert into Hall

(Зал, Всего\_мест, Всего\_рядов)

Values

(@Name, @Mest, @Rad)

end

alter proc insertProducer

@Name varchar(20),

@Surname varchar(20),

@Date varchar(11),

@From varchar(20)

as begin

insert into Producer

(Имя, Фамилия, Дата\_рождения, Страна)

values

(@Name, @Surname, @Date, @From)

end

alter proc insertStorehouse

@Name varchar(25),

@Kol int

as begin

insert into Storehouse

(Название, Количество)

values

(@Name, @Kol)

end

alter proc deleteEmployees

@ID int

as begin

delete from Employees where EmployeeID = @ID

delete from Orders where EmployeeID = @ID

end

alter proc deleteInfo

@id int

as begin

delete from FilmsInfo where FilmsInfoID = @id

delete from Films where FilmsIfoID = @id

end

alter proc deleteListPrice

@ID int

as begin

delete from PurchasingList where PurchasingListID = @ID

delete from Storehouse\_PurchasingList where PurchasingListID = @ID

end

alter proc deleteMenu

@ID int

as begin

delete from Food where FoodID = @ID

delete from OrderFood where FoodID = @ID

end

alter proc deleteSession

@ID int

as begin

delete from Films where FilmID = @ID

delete from OrderItself where FilmID = @ID

end

alter proc insertMenu

@ID int,

@Name varchar(35),

@Money money,

@MoneyCash money,

@Kol int

as begin

insert into Food

(StorehouseID, Название, Цена, Себестоимость, Количество)

values

(@ID, @Name, @Money, @MoneyCash, @Kol)

end

alter proc insertSession

@IDF int,

@IDH int,

@Money int,

@Kol int,

@Date date,

@Time time(0)

as begin

insert into Films

(FilmsIfoID, HallID, Цена\_билета, Билеты, Дата, Время)

values

(@IDF, @IDH, @Money, @Kol, @Date, @Time)

end

alter proc insertInfo

@IDC int,

@IDG int,

@IDP int,

@From varchar(20),

@Name varchar(25),

@Date date,

@Time time(0),

@Valua float

as begin

insert into FilmsInfo

(MoviesSalesmanID, GenreID, ProducerID, Страна, Название, Дата\_выпуска, Длительность, Оценка)

values

(@IDC, @IDG, @IDP, @From, @Name, @Date, @Time, @Valua)

end

alter proc insertListPrice

@IDS int,

@Name varchar(25),

@Kol int

as begin

insert into PurchasingList

(SupplierID, Товар, Количество)

values

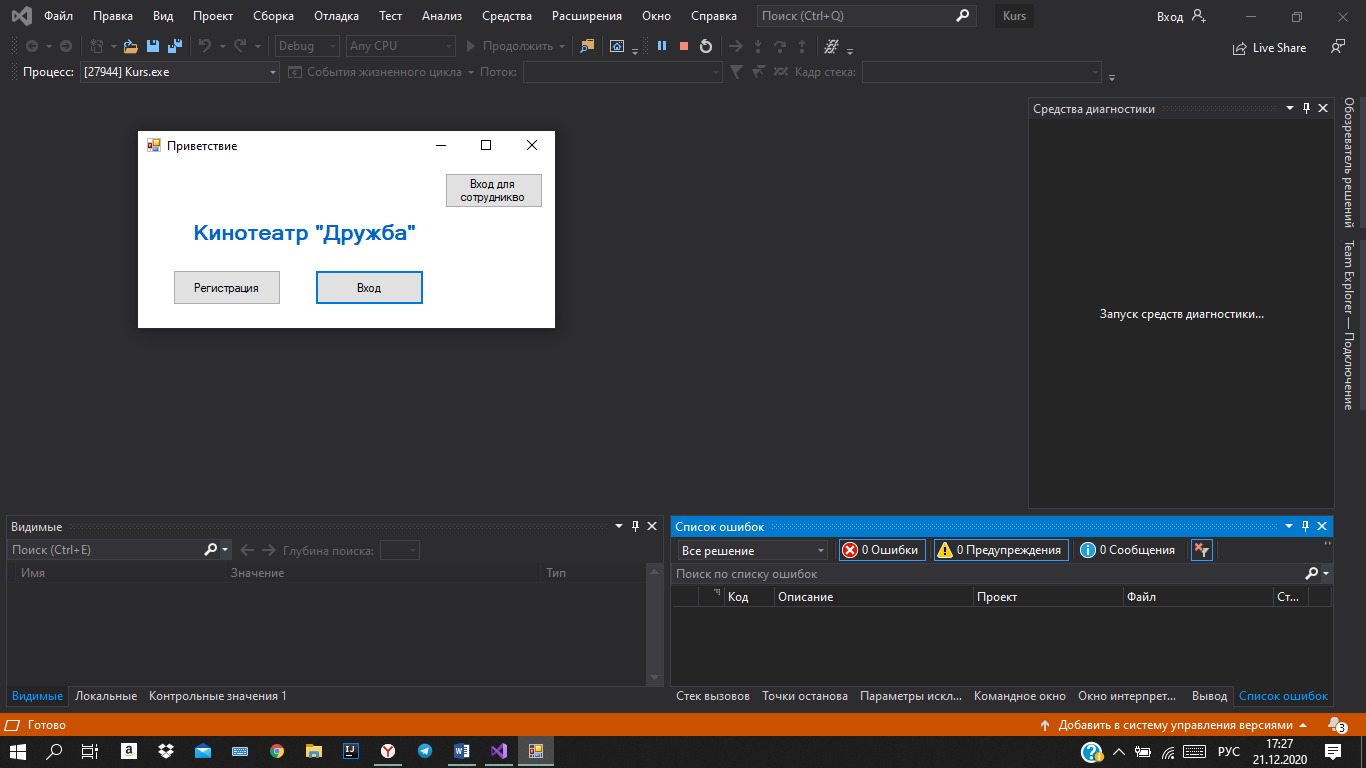
(@IDS, @Name, @Kol)

end

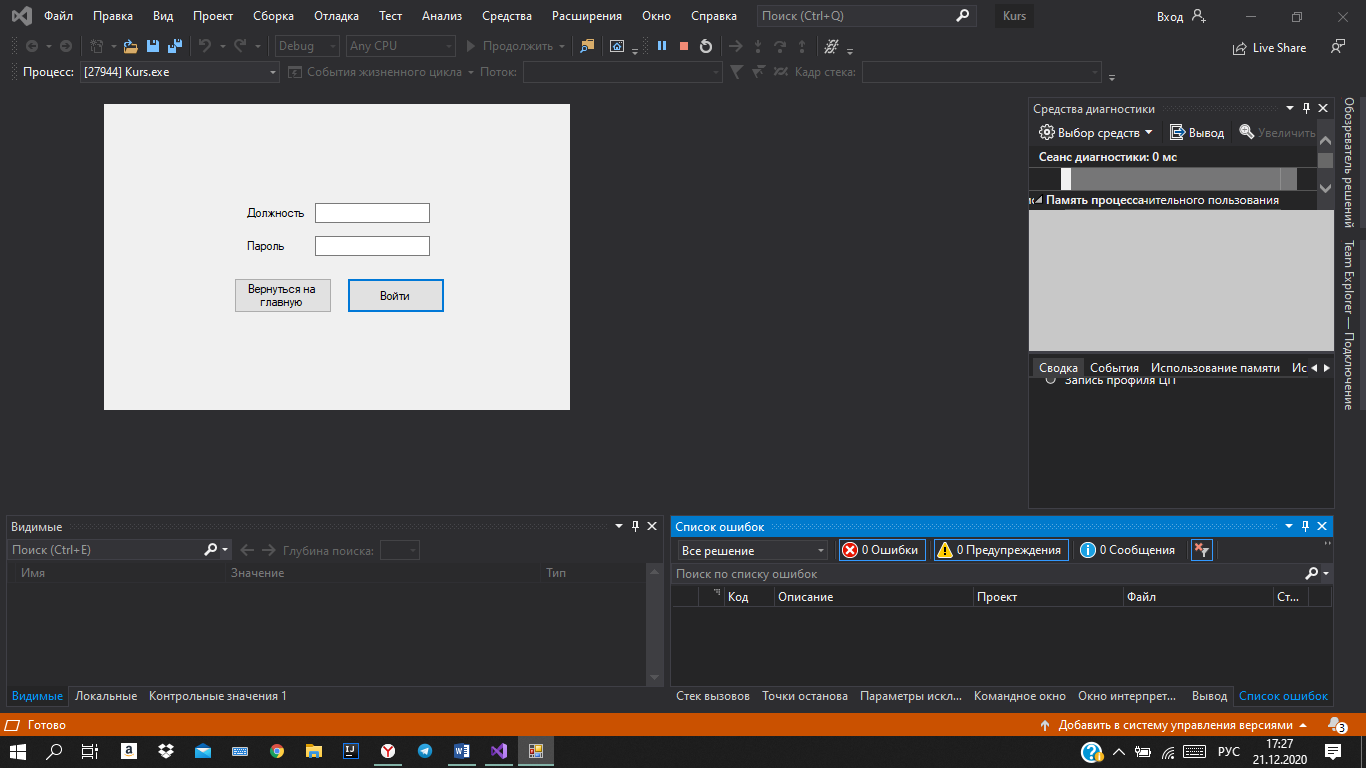
Клиентская часть

Для упрощения взаимодействия с сервером на языке C# было создано клиентское приложение, реализующее описанный функционал.

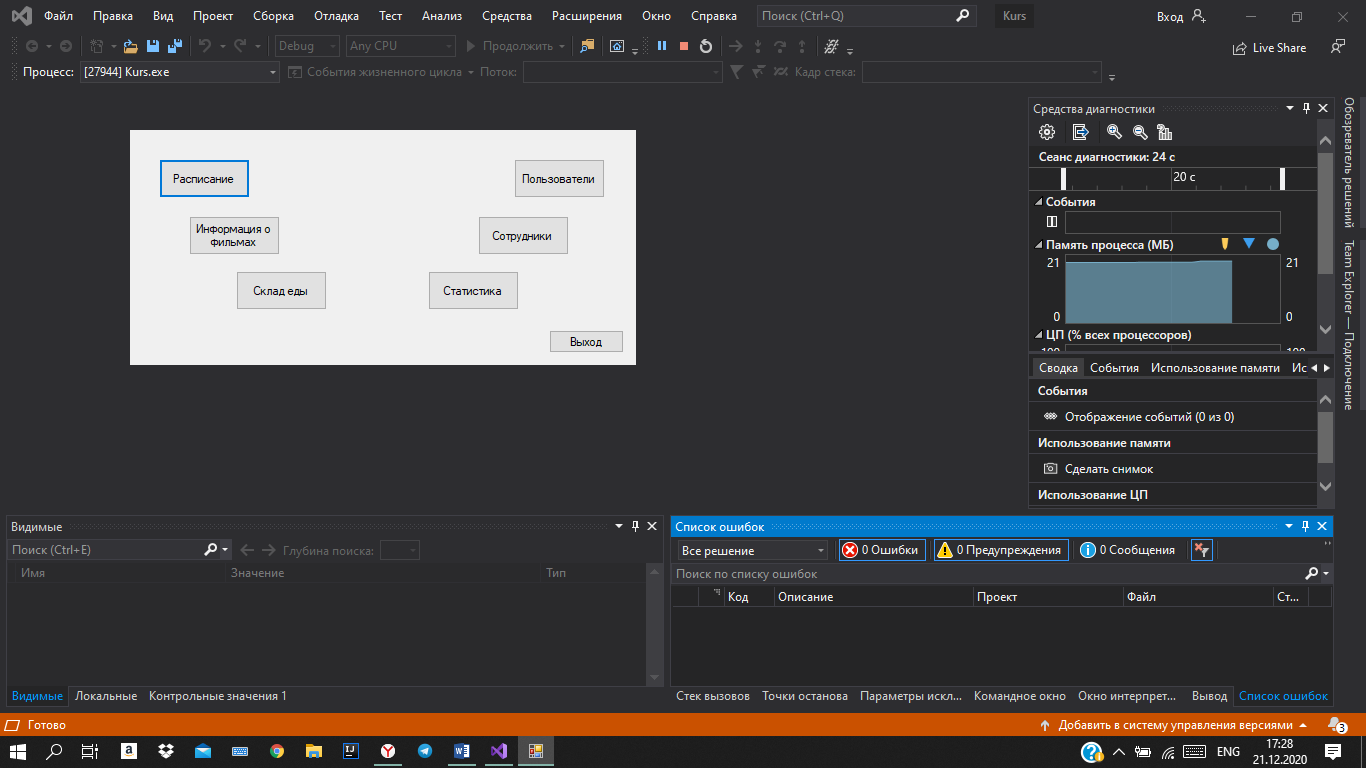
Главная экран



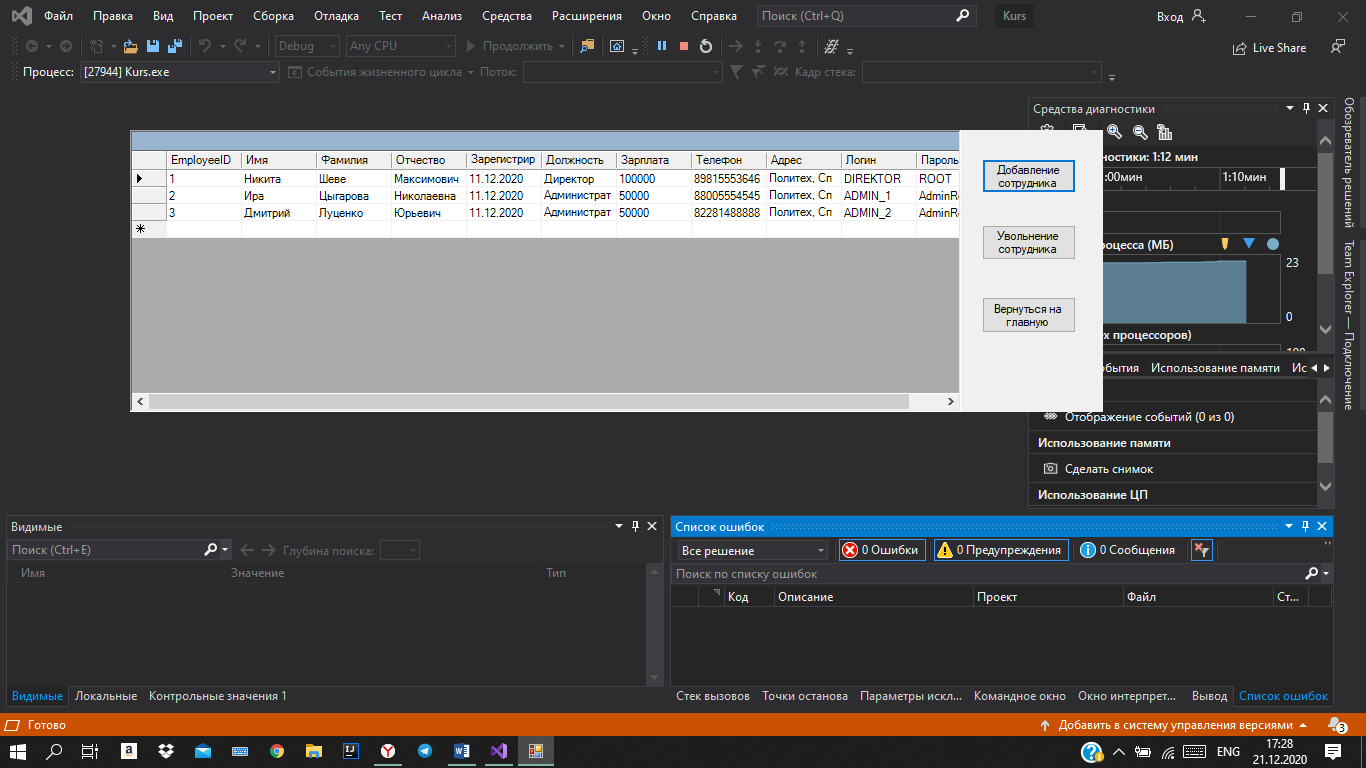
Вход для сотрудников

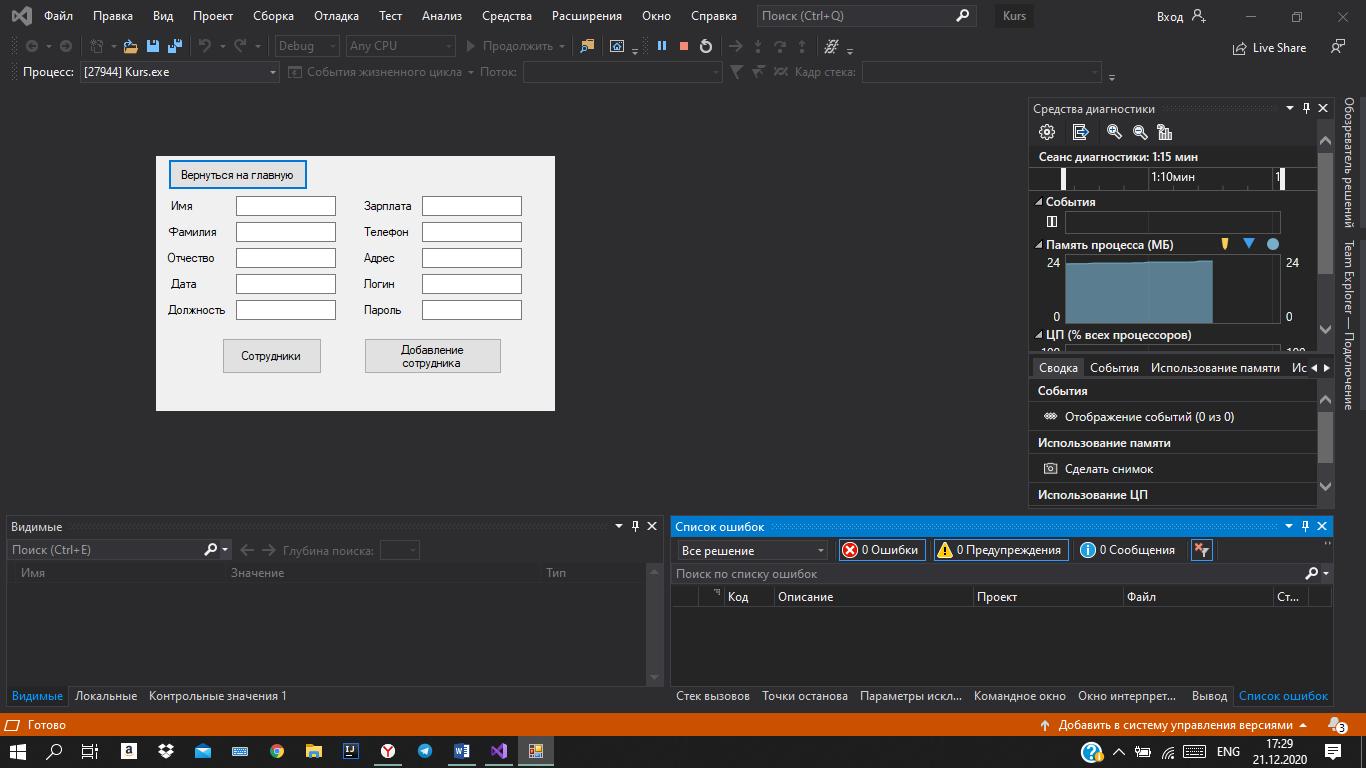


Главный экран Директора

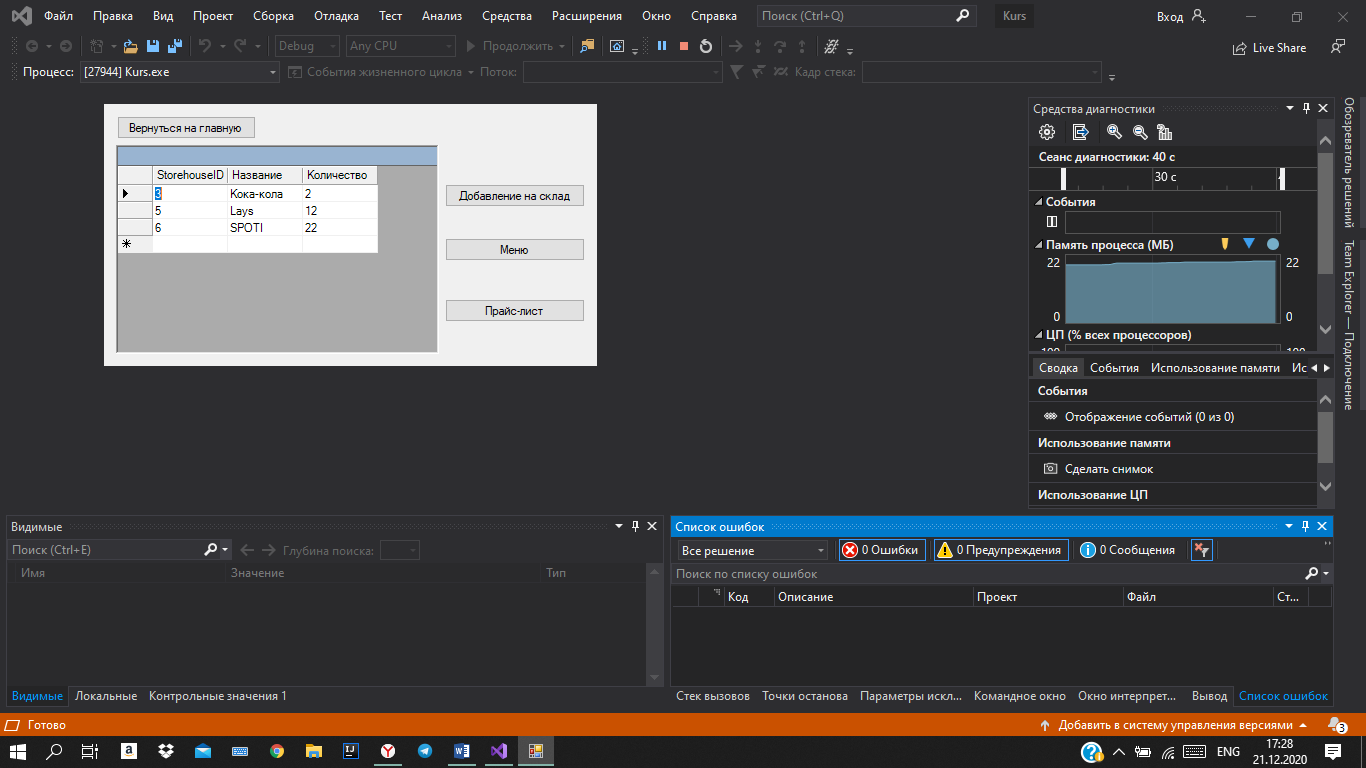


Добавление сотрудника

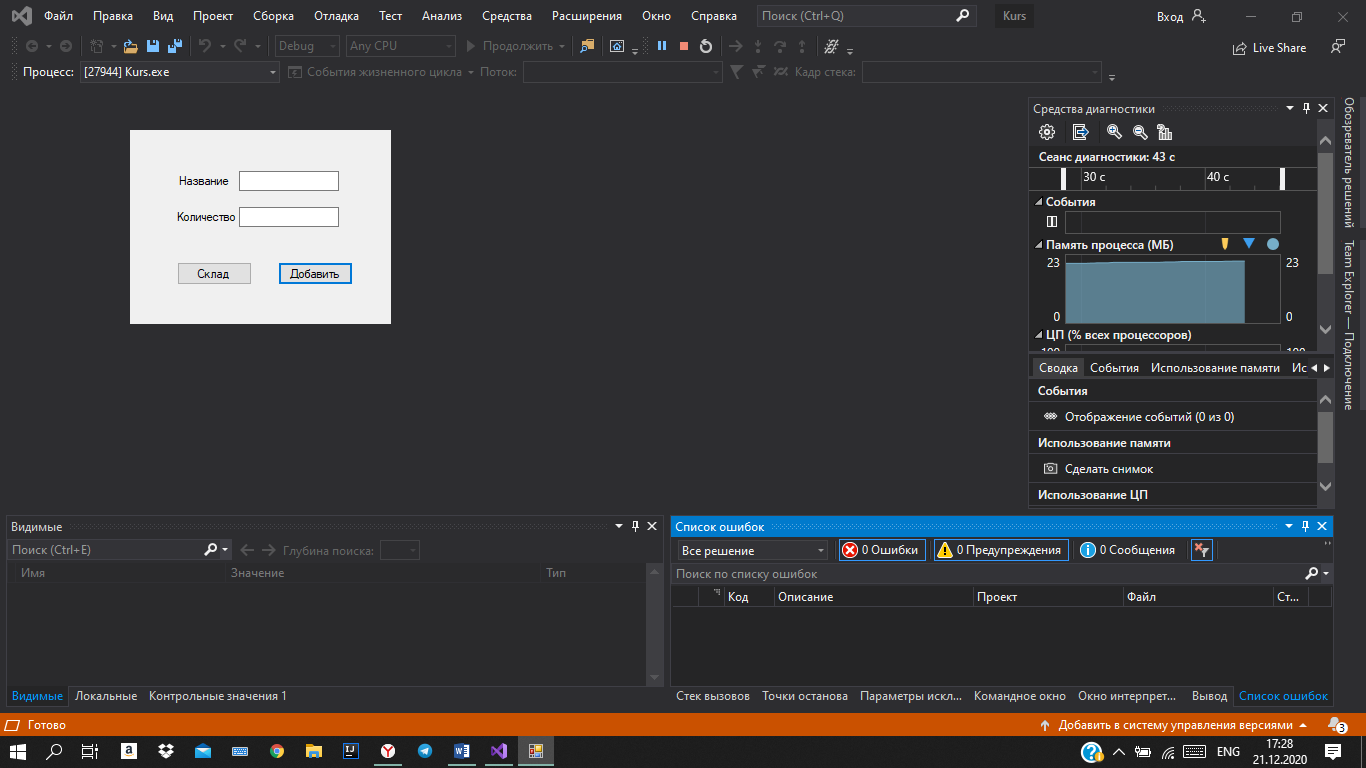




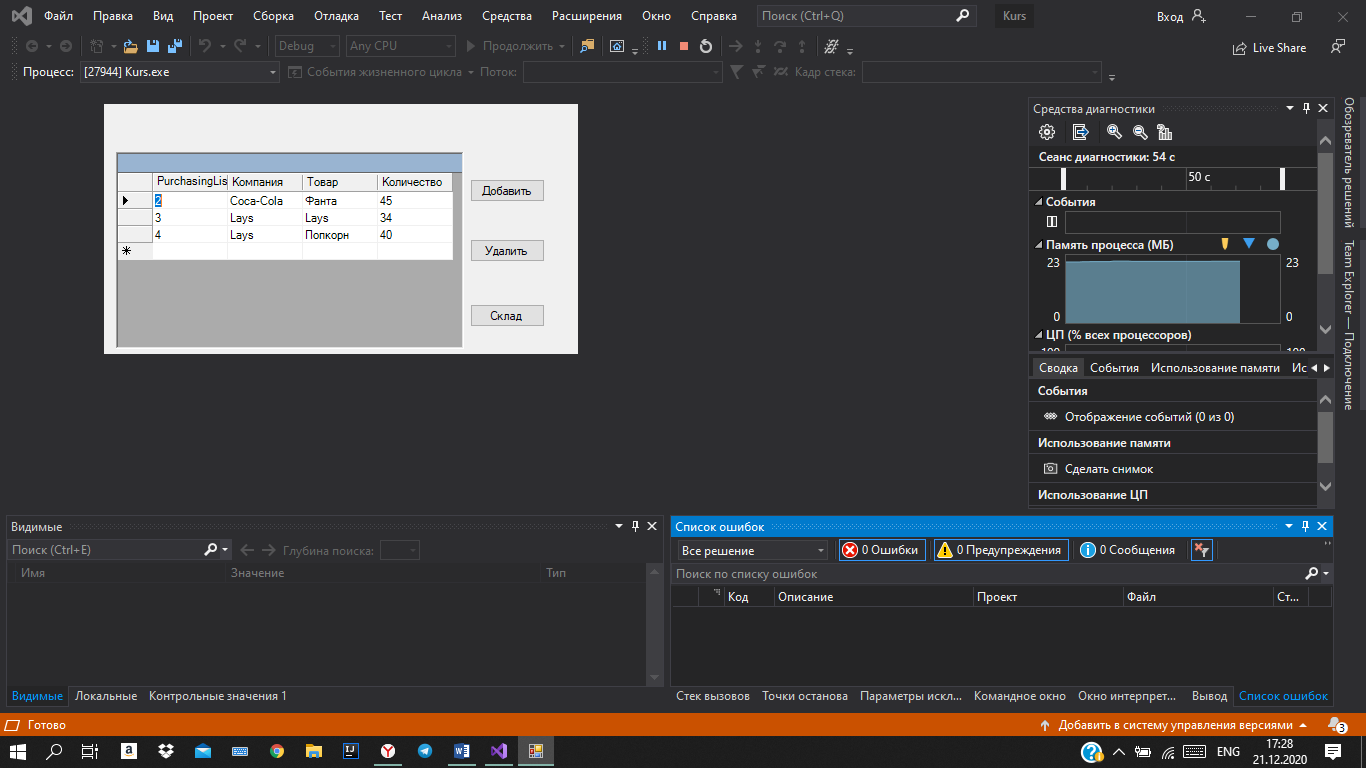
Просмотр склада



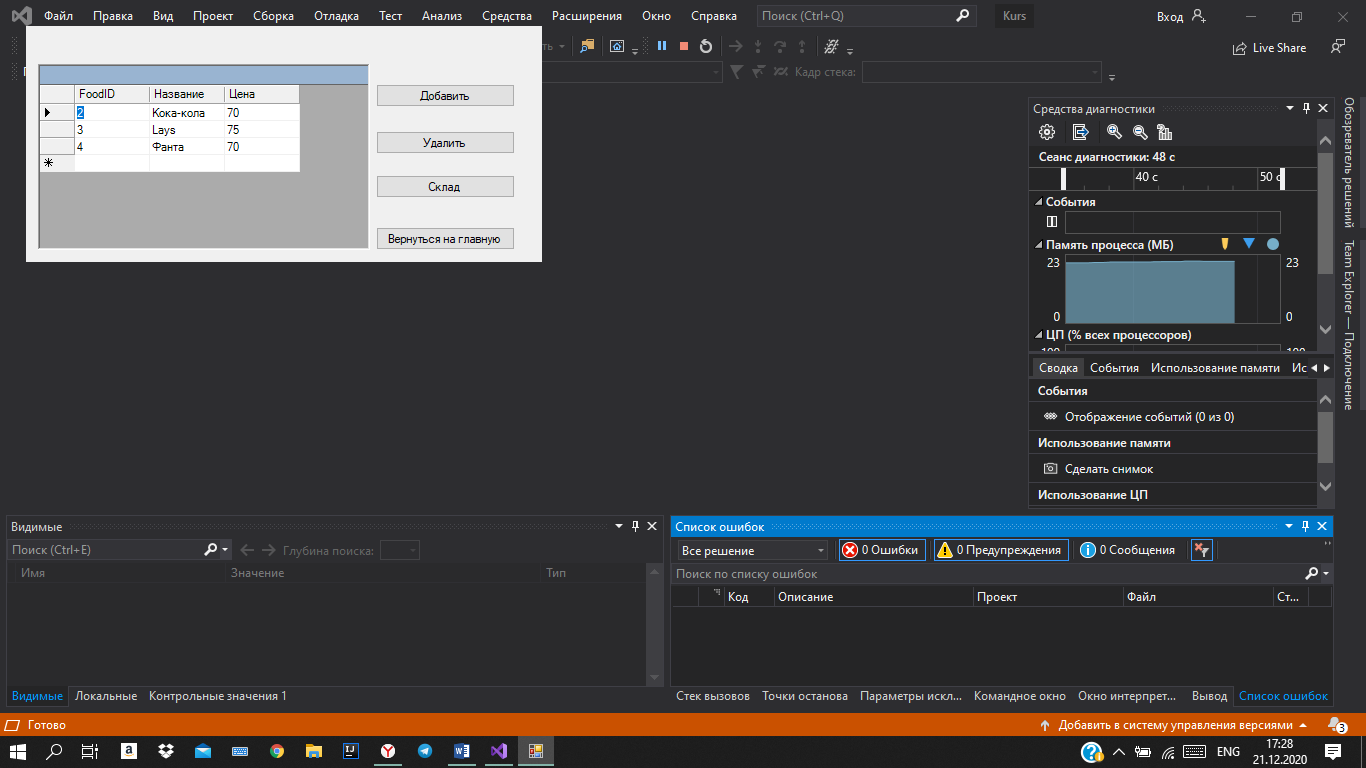
Добавление на склад



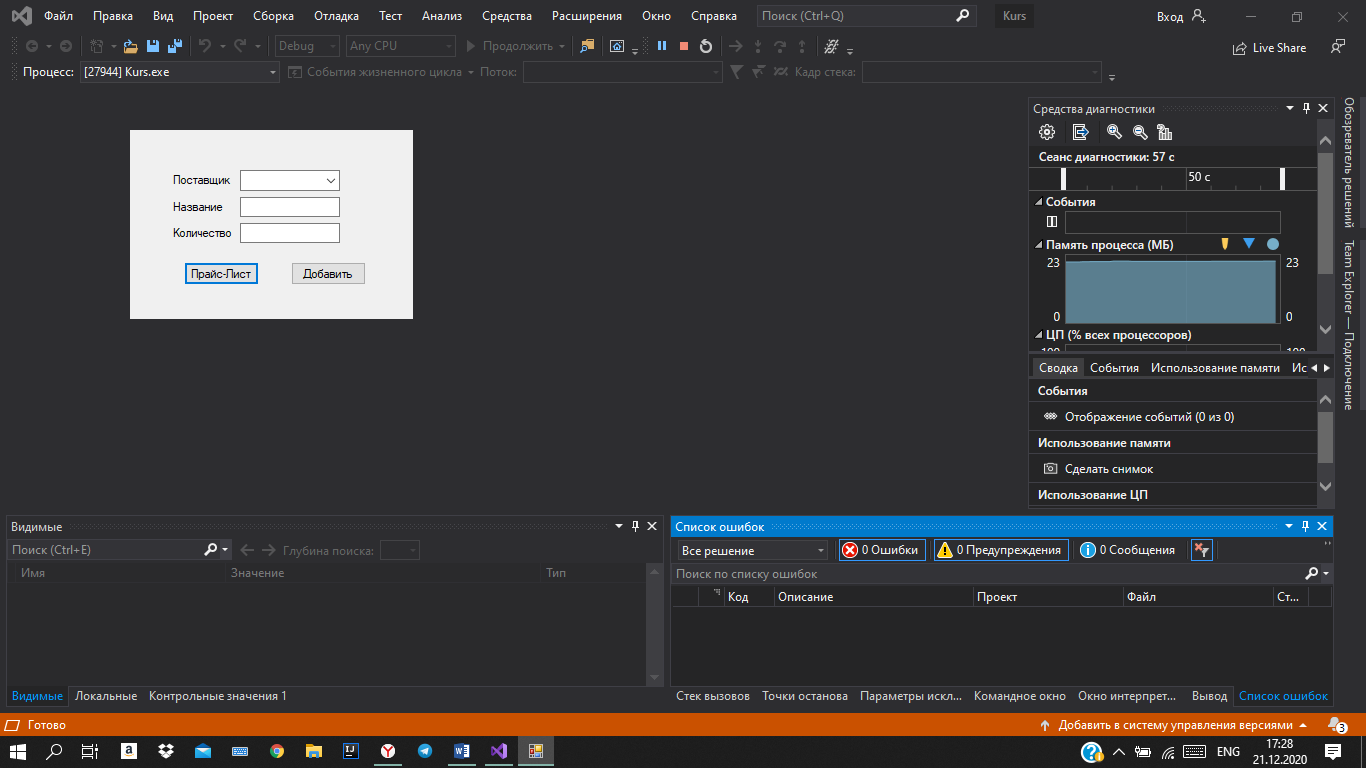
Склад



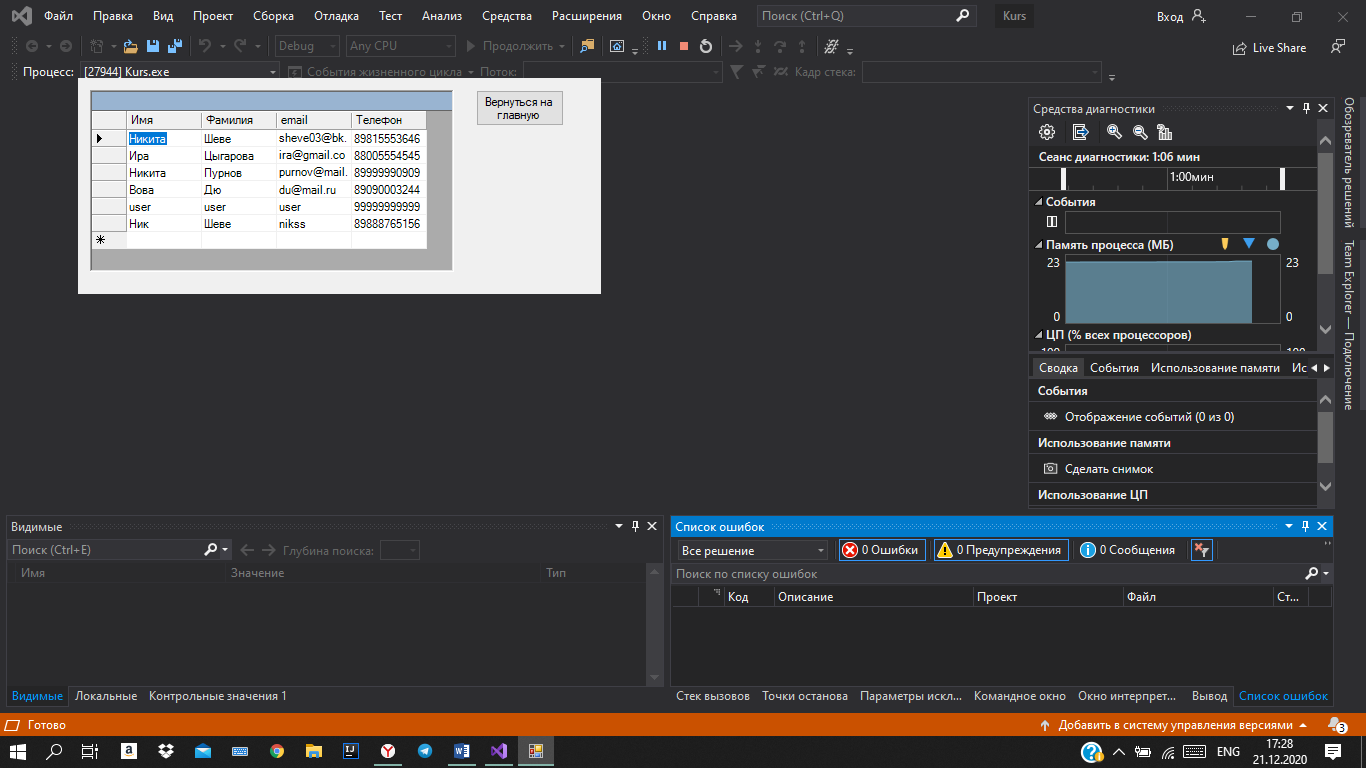
Меню



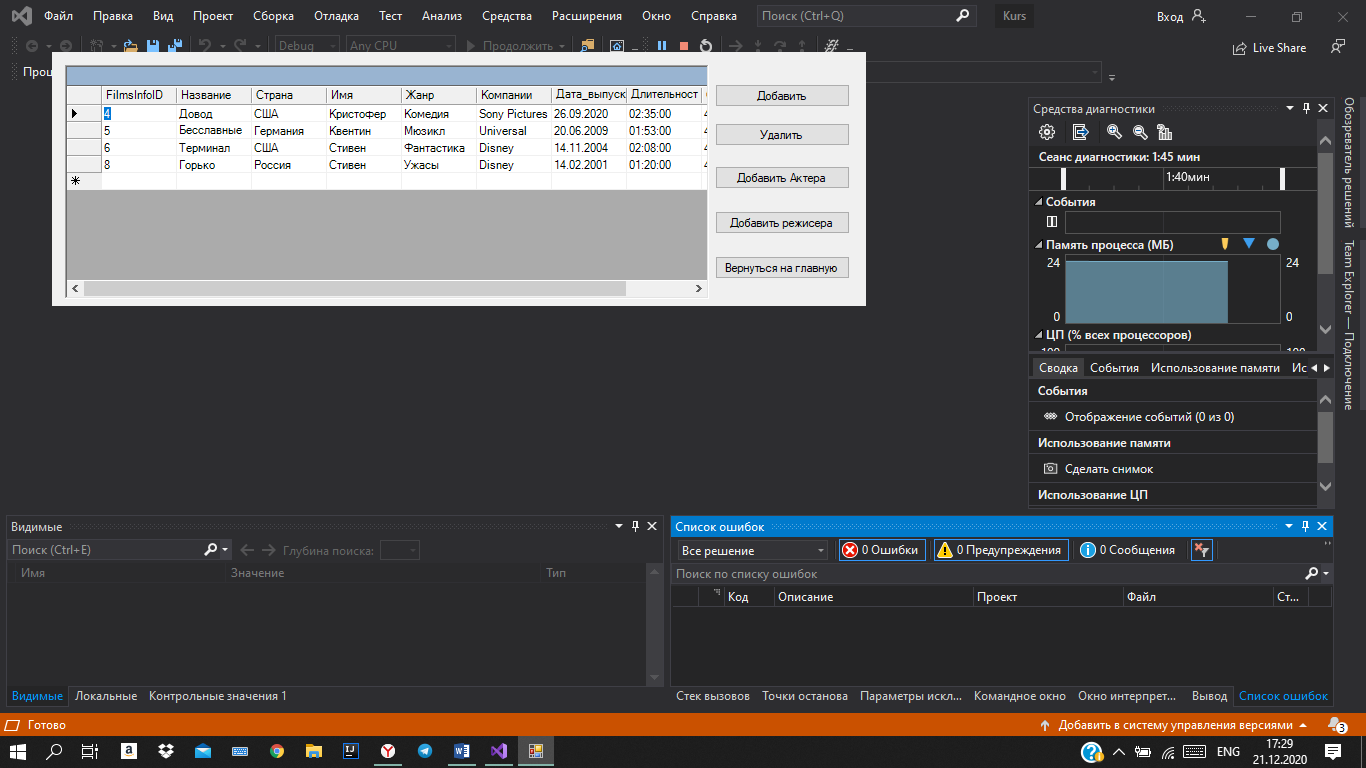
Добавление в Прайс-Лист



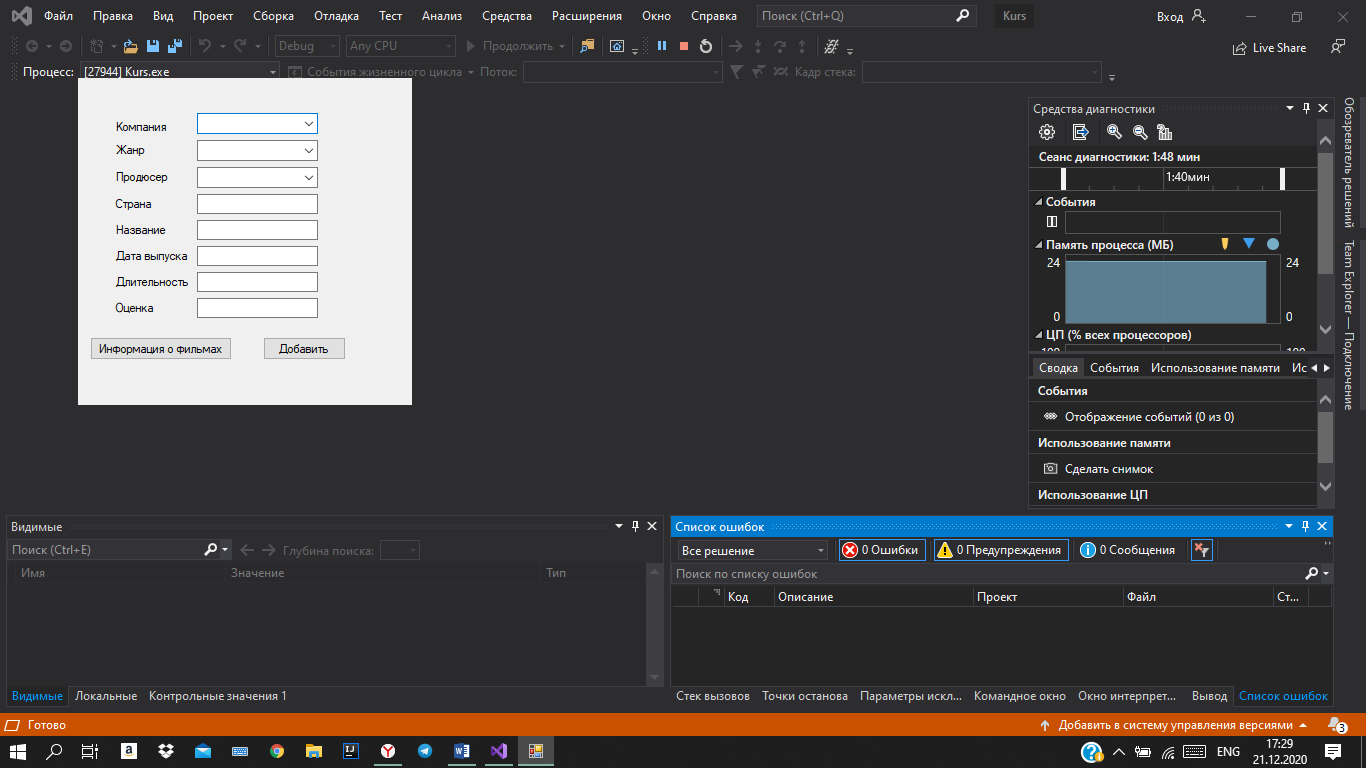
Просмотр клиентов



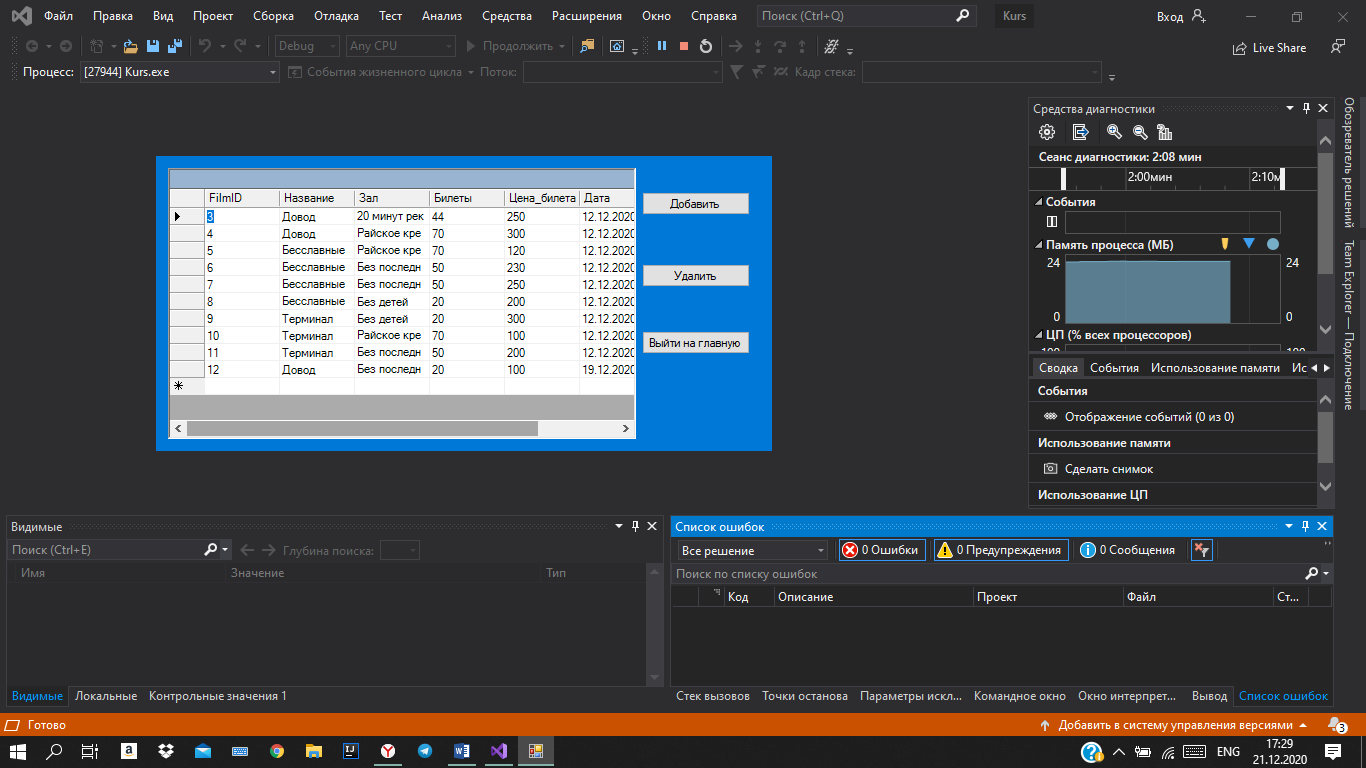
Просмотр информации фильмов



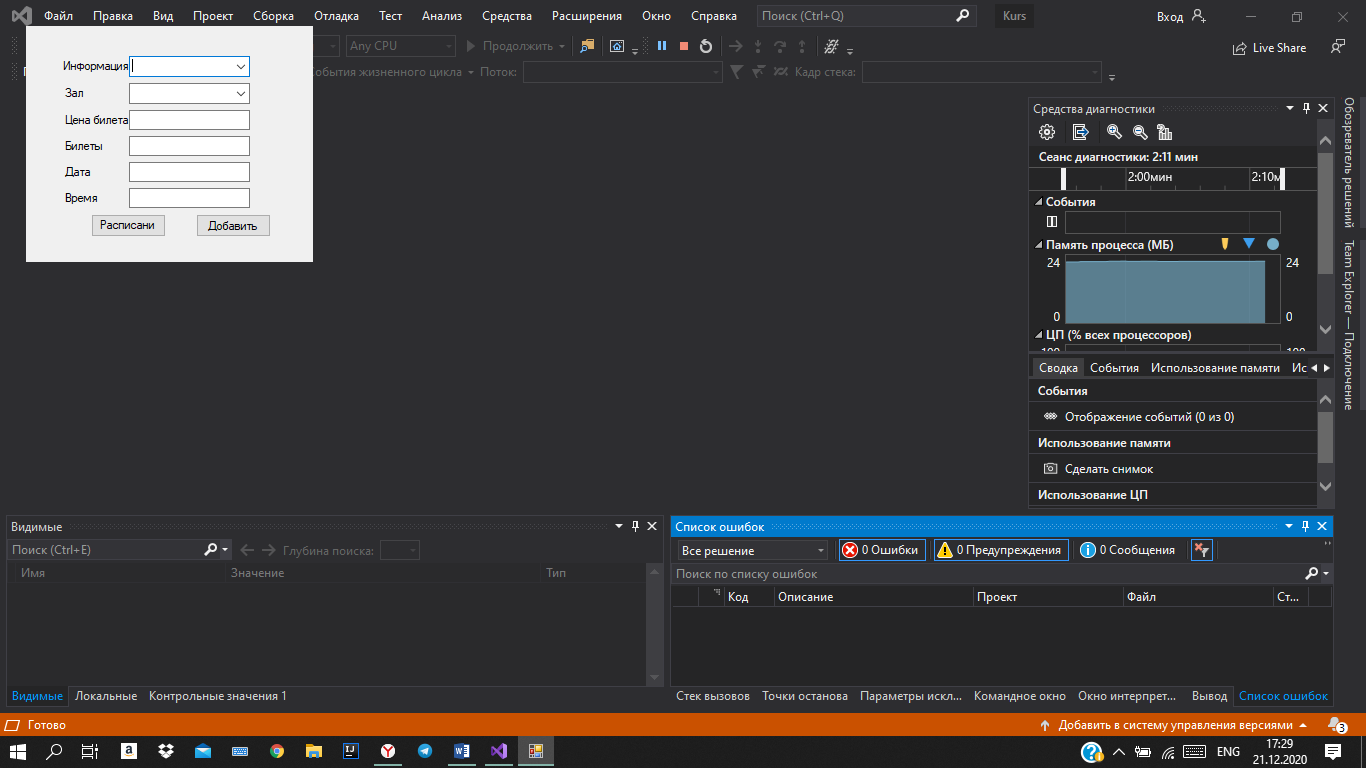
Добавление информации фильмов



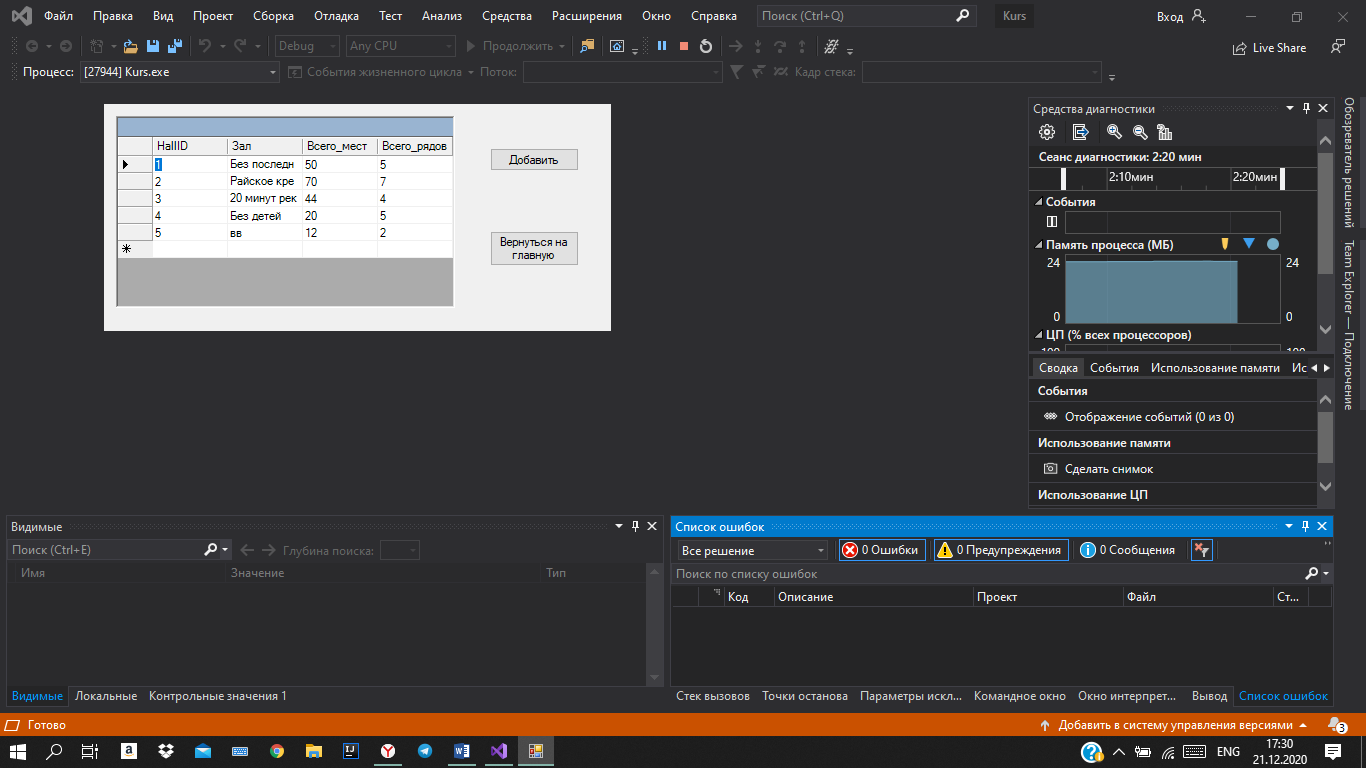
Просмотр сеансов



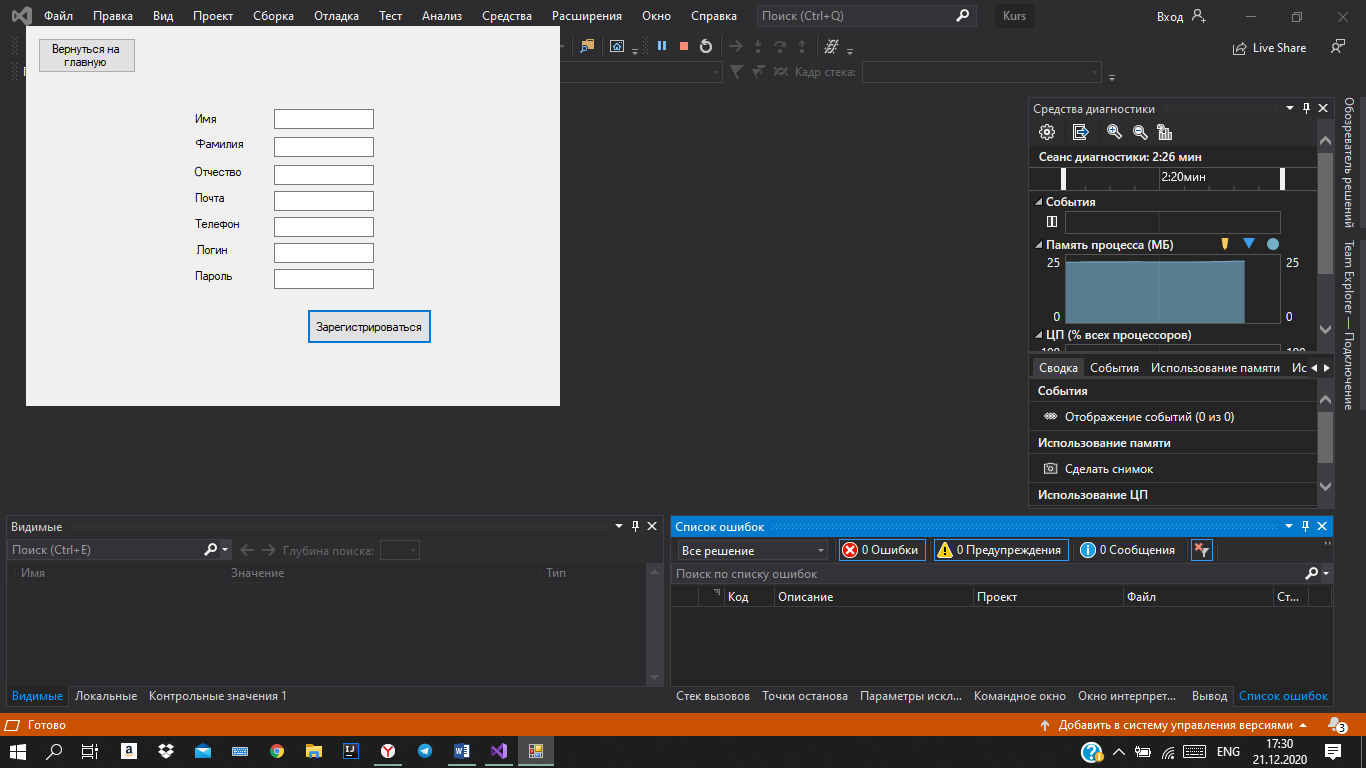
Добавление сеансов



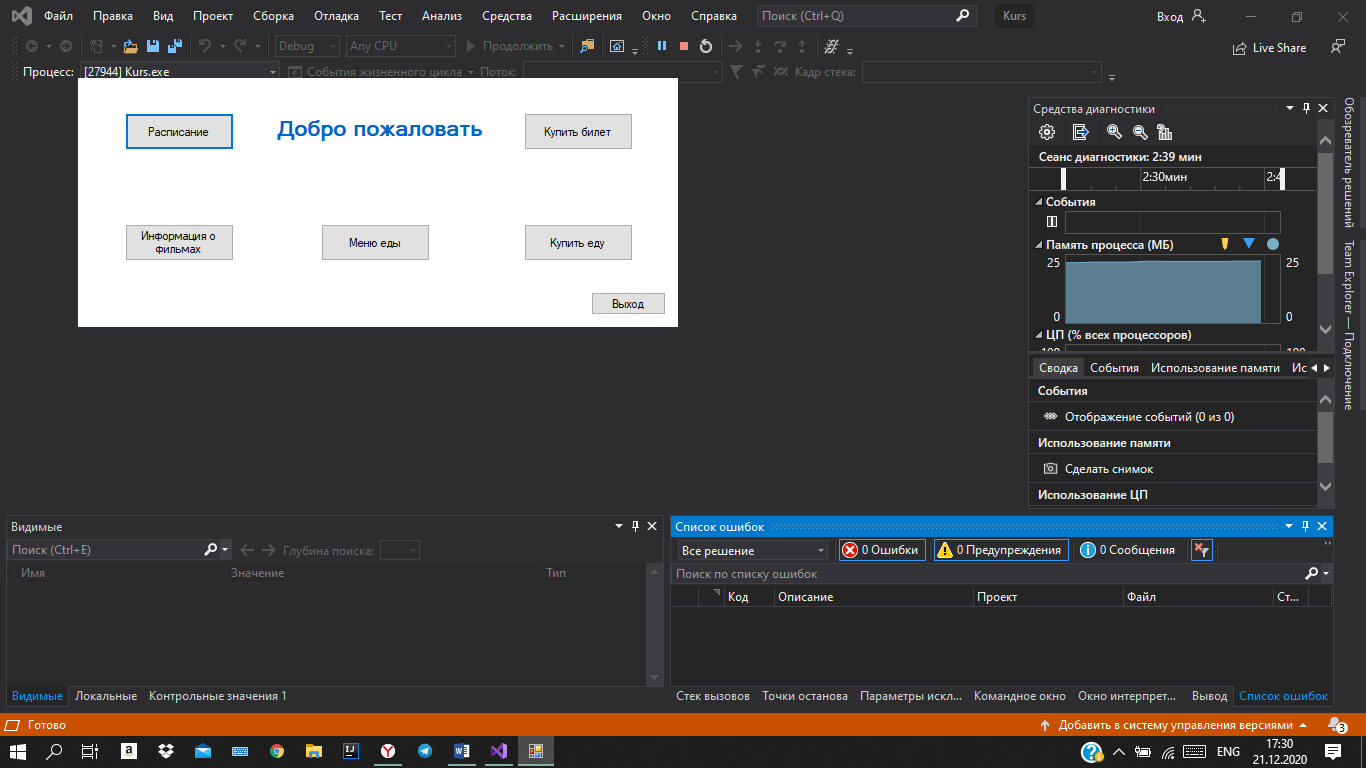
Просмотр залов



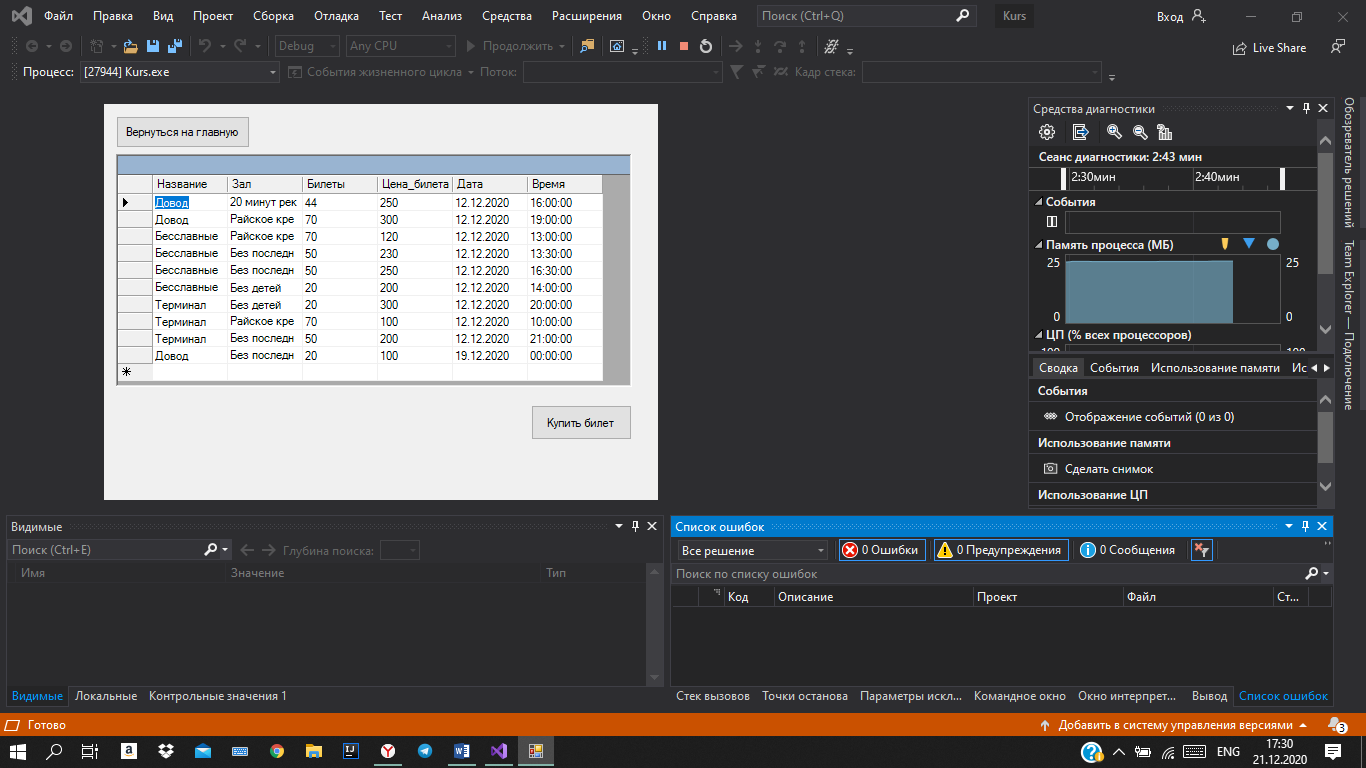
Экран регистрации клиента



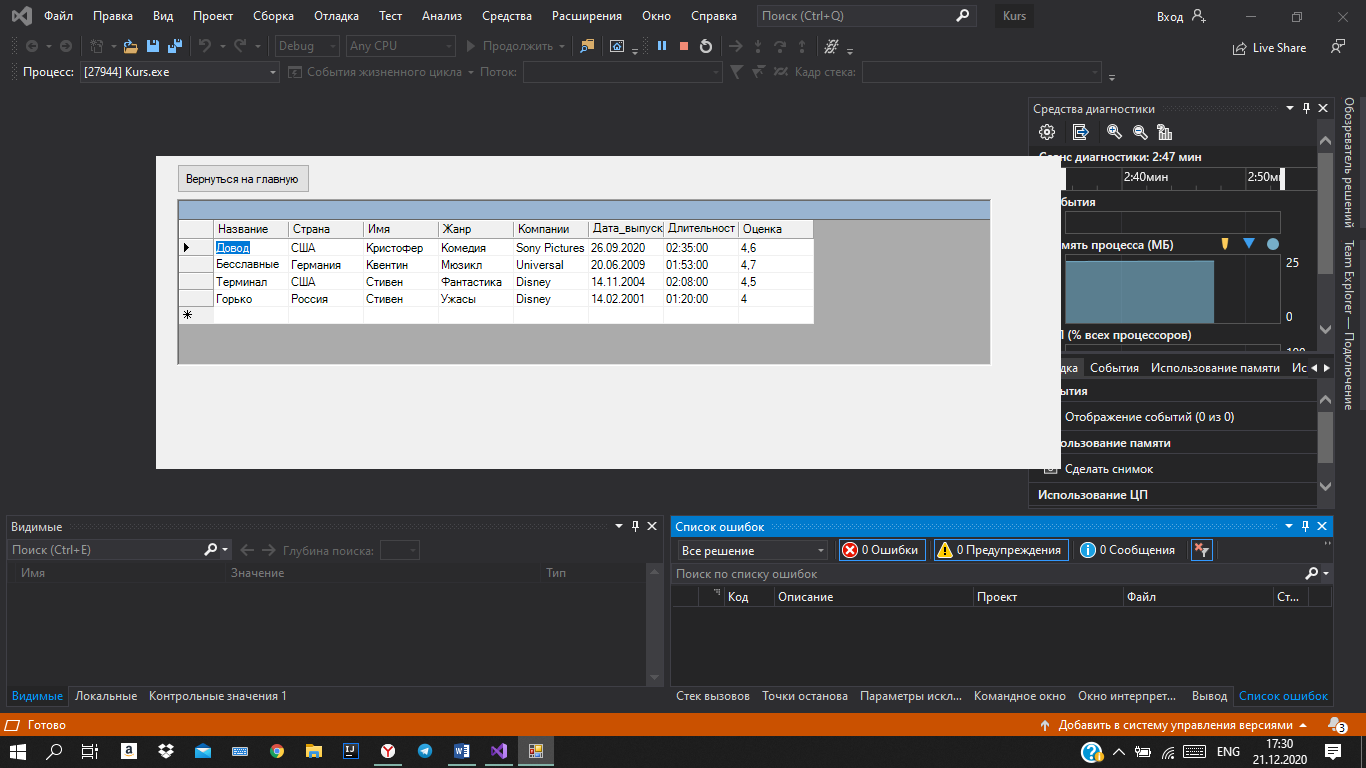
Главный экран клиента



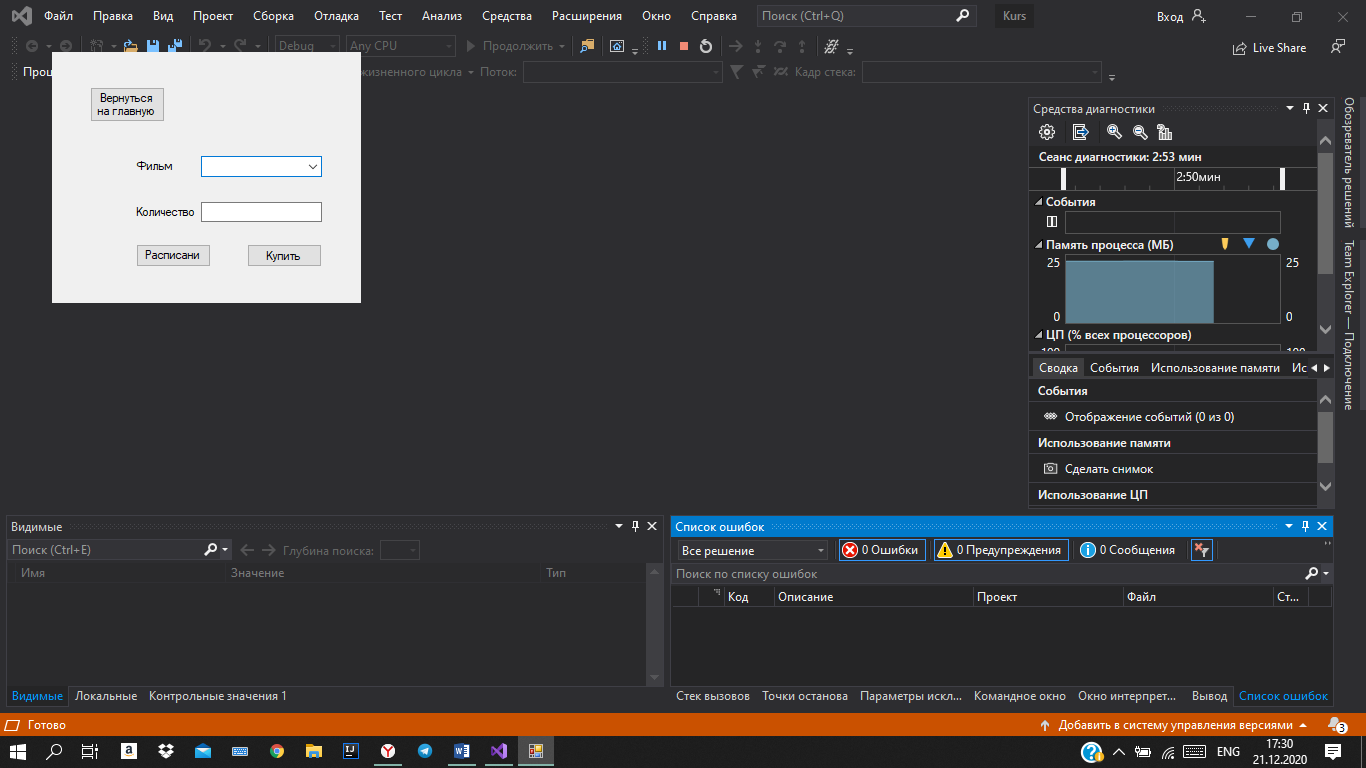
Просмотр сеансов



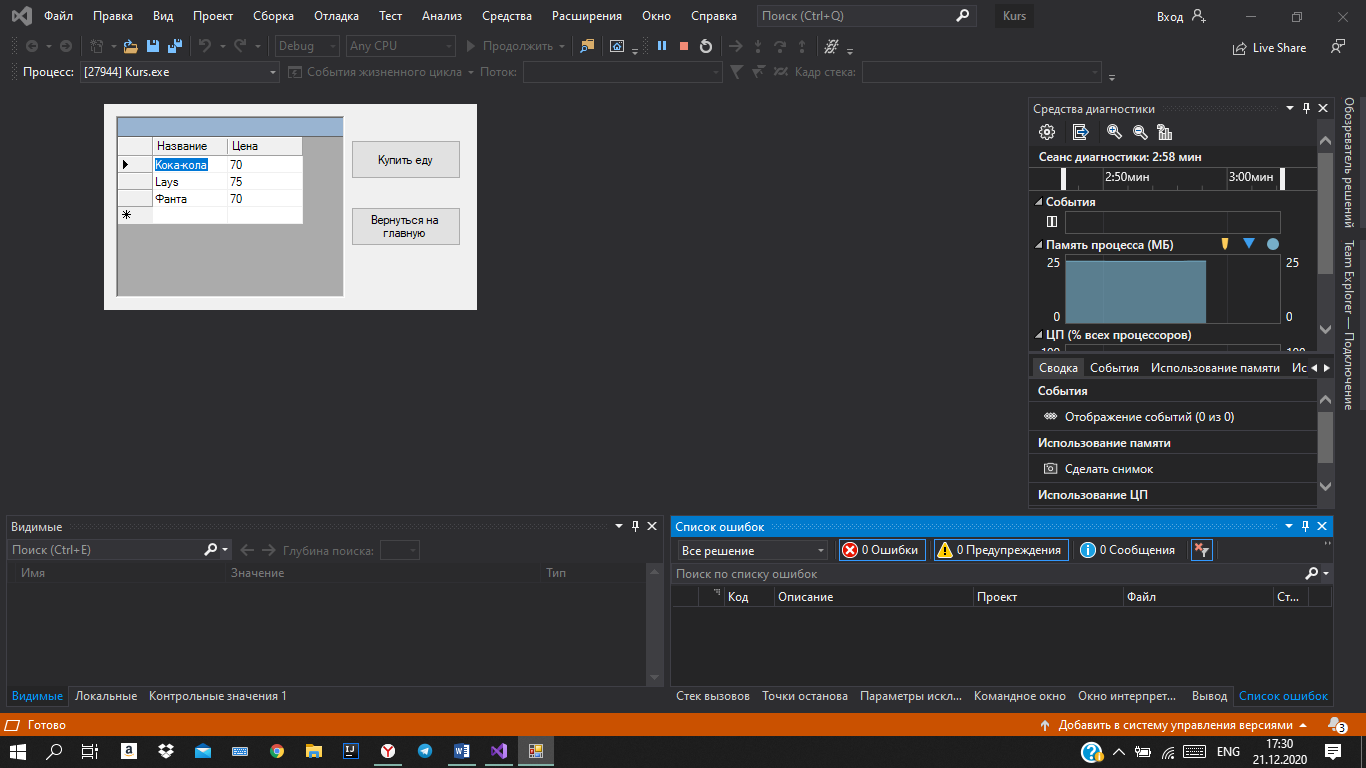
Просмотр информации о фильмах



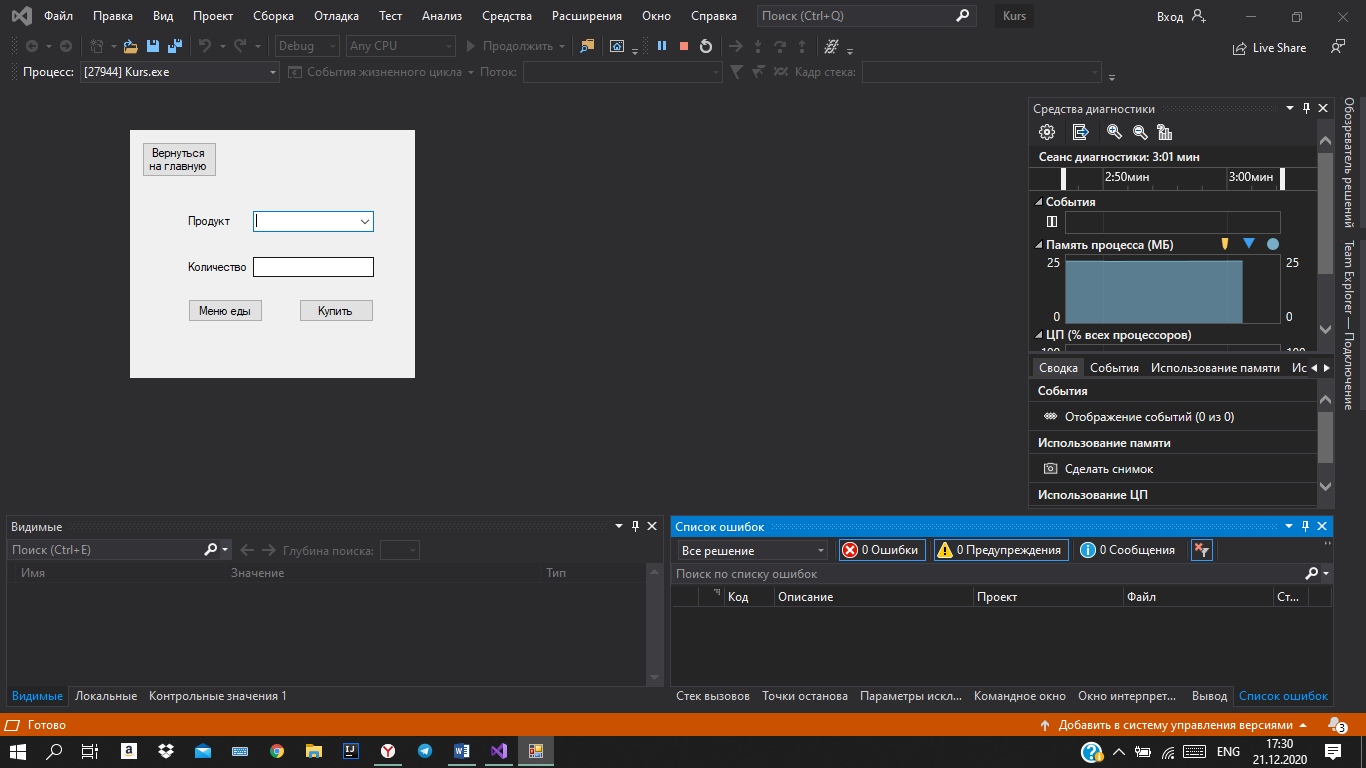
Купить билетов



Просмотр меню



Купить товар



Вывод

В ходе курсовой работы была реализована система для работы ресторана. Для её реализации была использована MSSQLSMS. Клиентская часть выполнена с помощью платформы Visual Studio 2019 на языке C#.

Список литературы

1.https://visualstudio.microsoft.com/ru/?rr=https%3A%2F%2Fyandex.ru%2F - платформа

Visual Studio 2019

2.https://professorweb.ru/my/sql-server/2012/level3/3\_18.php

3.https://docs.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-2019