

**Технически Университет Варна**

Факултет по изчислителна техника и автоматизация

Курсов Проект

**Приложение за видеотека**

Изготвили:

Иван Антонов Михов Ф.Н: 17621648 Преподавател:

Ас. Николаева

Дисциплина:

Специалност: СИТ Курс: III Група: 2 Б ТСП

1. Описание на заданието
   1. Изискване към системата за Видеотека – общите изисквания към системата са
      1. Изискване за регистрация на потребителя като се осигури защита на неговите данни
      2. Изискване за вход на потребител в системата
      3. Изискване потребителят да може да разглежда, търси и добавя филми или сериали в неговото хранилище
      4. Изискване за обвързване на потребителят към абонаментен план, които да разрешава на потребителя да изпълнява отделни функции
      5. Изискване на администратора за служебен вход като се използва неговото ID за вход
      6. Изискване към администраторският вход за
         1. Добавяне на филми или сериали
         2. Добавяне на жанрове
         3. Добавяне на други служители
         4. Добавяне на други позиции
         5. Добавяне на други типове
2. Описание на функционалните изисквания
   * 1. Модул за регистрация на потребител

* Потребителят въвежда потребителско име, парола и имейл
* Ако не съществува такъв потребител със същите данни той се вкарва базата данни като неговот потребителско име и парола се хешират така не се работи със тяхните реални данни
* Потребителят трябва да попълни всички полета за да бъде регистриран
  + 1. Модул за вход на потребител в системата

- Потребителят въвежда потребителското си име и парола, при съвпадение той се допуска в системата като се зареждат неговите данни

- Потребителят трябва да въведе всички полета за да бъде успешна операцията за вход

* + 1. Модул за вход в системата като администратор

- Администраторът или служителят може да влезе в системата като въведе съответните данни в потребителското име се въведе – admin и за парола EMP и съответното ID на служителят, които има достъп в системата

- Администраторският вход в системата позволява на служителя да прави следните функции

- Добавяне и промяна на градовете на където има офиси на тази система

- Добавяне и промяна на филмите и сериалите във системата

- Добавяне и промяна на абонаментните планове, които потребителите могат да избират за да използват различните плюсове и минуси

- Добавяне и промяна на жанровете налични на потребителя да търси филми или сериали по тях

- Добавяне и промяна на служители като могат да се добавят като се въвеждат данните на служителя неговите имена, телефоннен номер, заплата, позиция и град

* + 1. Модул за вход в системата като потребител

- Потребителят влиза във системата ако вече съществува в системата

- Ако за първи път е влиза в системата той трябва да си избере абонаментен план за да може да използва системата, като избере начална дата. Този абонаментен план е със продължителност 1 месец. След това потребителя може да си закупи друг абонамент. Целта на този план и на тази система е на всеки 1 месец потребителят да има възможността да не плаща или да смени абонаментният си план

- Потребителят не може да избира начална дата за стартиране на абонаментният план по стара от настоящата

- След като закупи този абонаментен план потребителят има достъп до всички функции на системата

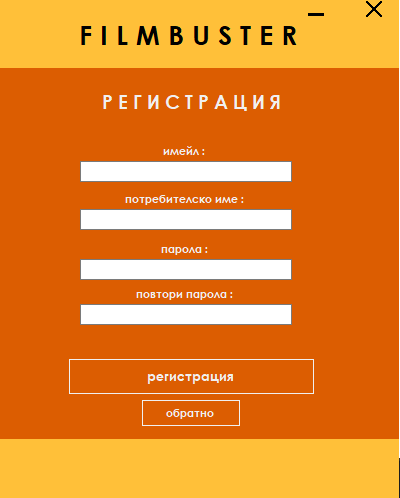
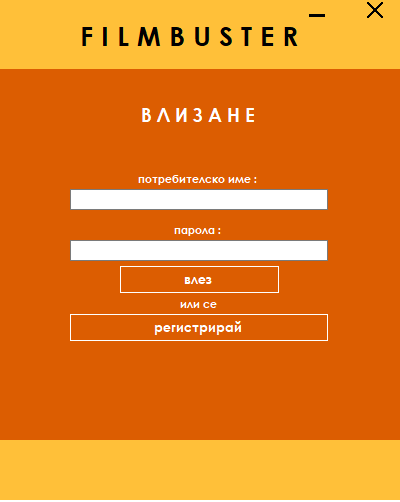
- Потребителят може да търси филми или сериали по избран от него филтър, които запаметява или може да търси всички налични филми или сериали

- Потребителят може да добавя филми или сериали във любими, като това е неговото лично хранилище, което е индивидуално за всеки потребител. Това хранилище се зарежда при вход на потребителя в системата

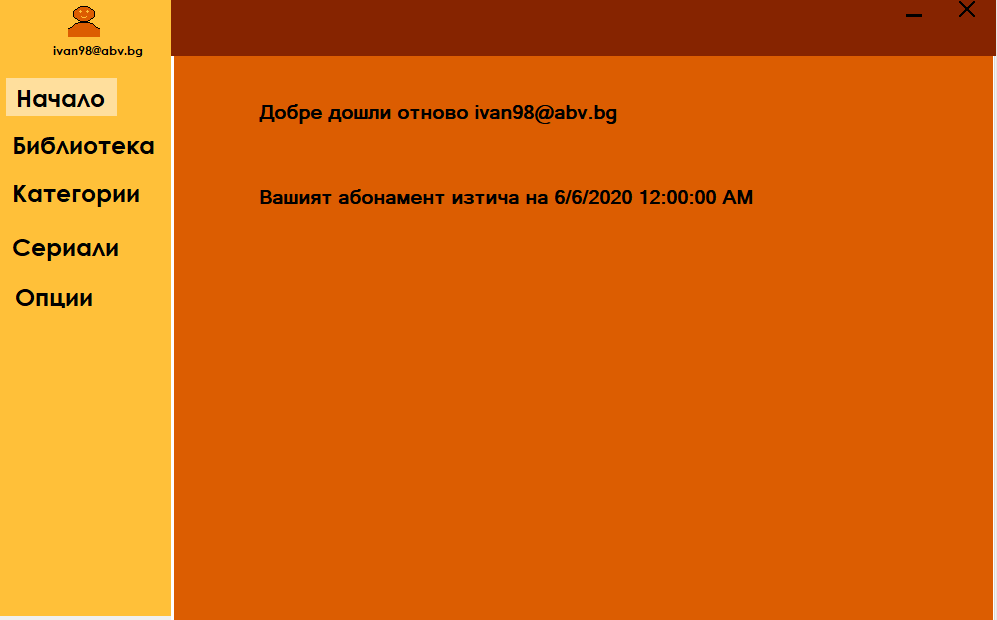
- Потребителят също така може да премахва филми или сериали от неговото лично хранилище

1. Основни екрани и форми

Екран на регистрация и влизане в системата

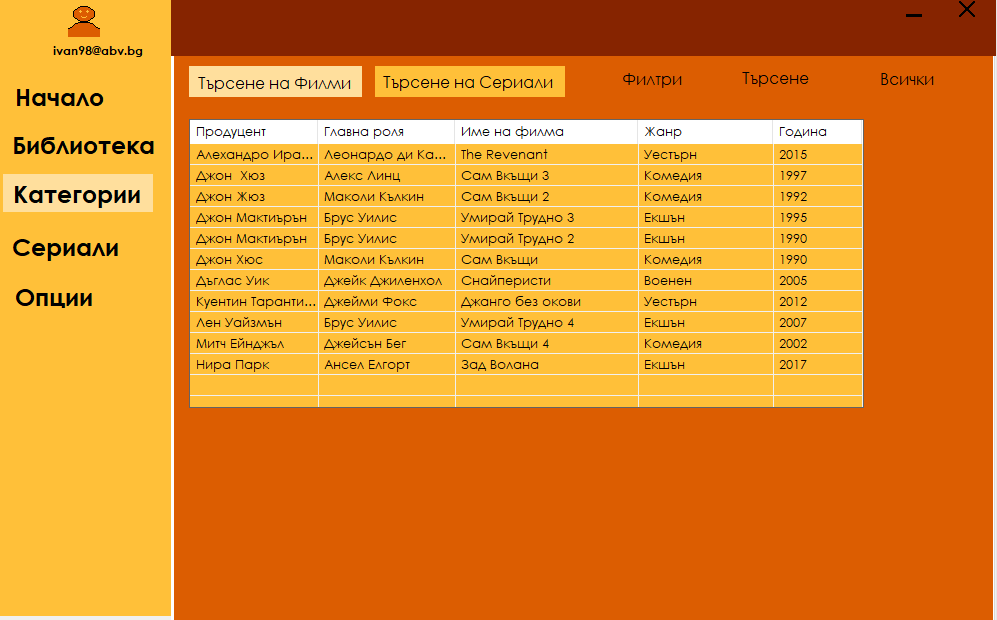
Основен екран при влизане в системата в потребителски режим

****

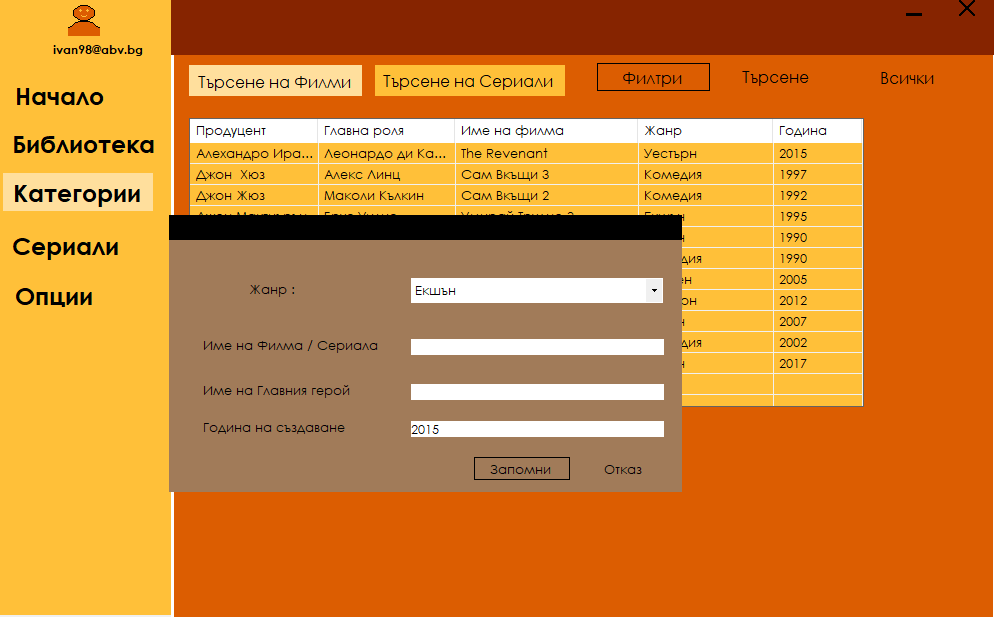
Екран показващ библиотеката от любими филми за този потребител

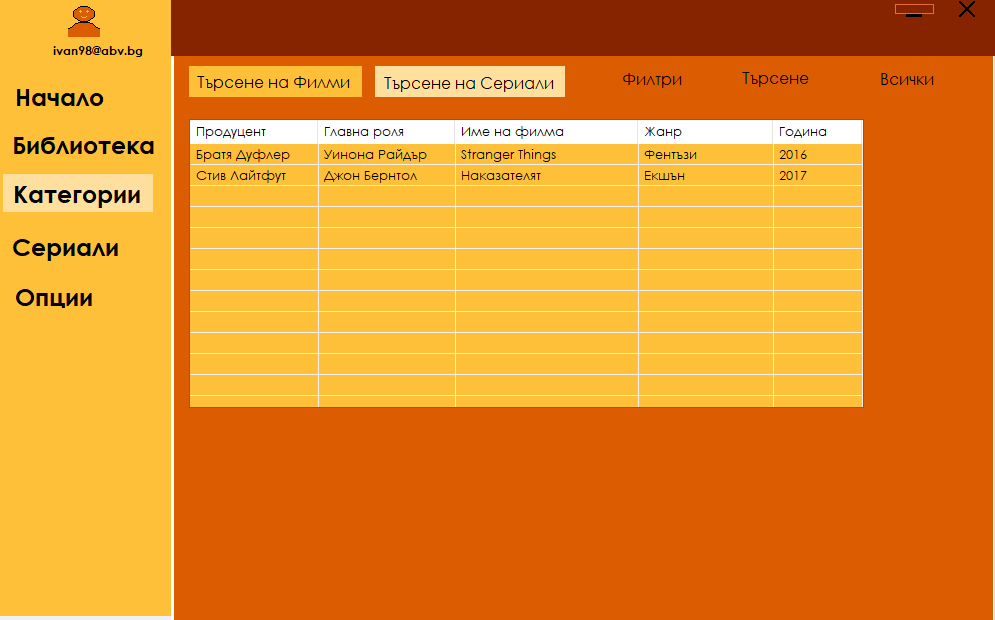


Екран показващ всички налични филми без приложен филтър за търсене



Екран показващ попълването на филтъра за търсене спрямо избраният радио бутон / филми или сериали. Не е задължително формата да е попълнена за да изведе резултат.

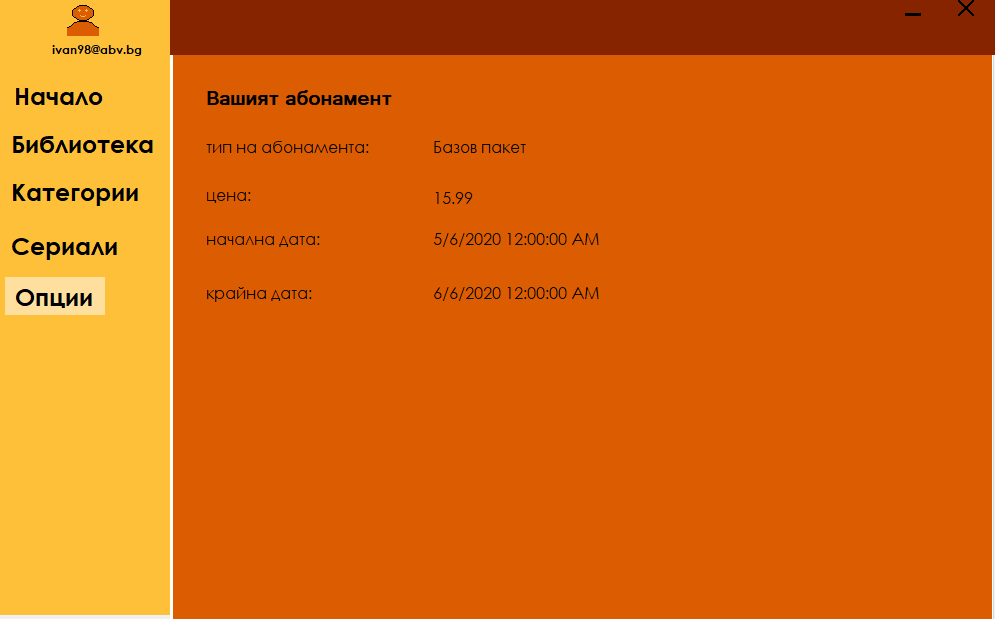


Екран показващ всички налични сериали без приложен филтър

Екран показващ библиотеката от любими сериали за този потребител



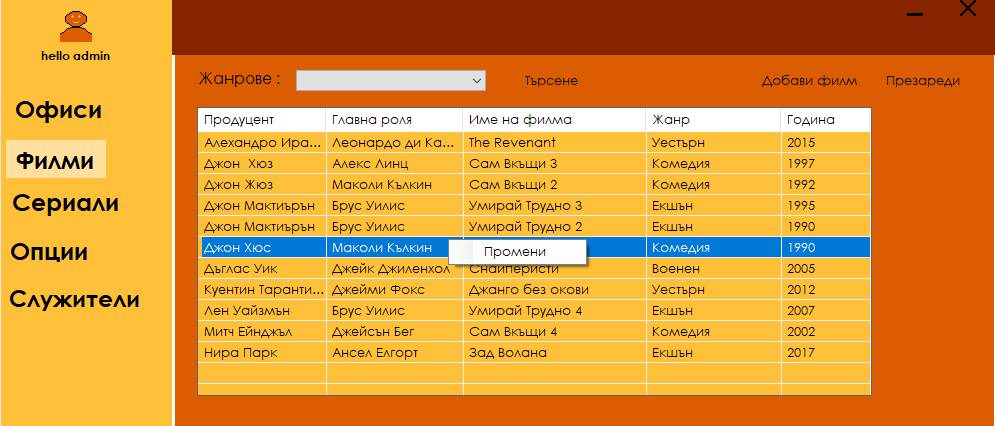
Екран показващ цялата информация за абонамента на потребителя, като абонаментен план, цена, начлна и крайна дата



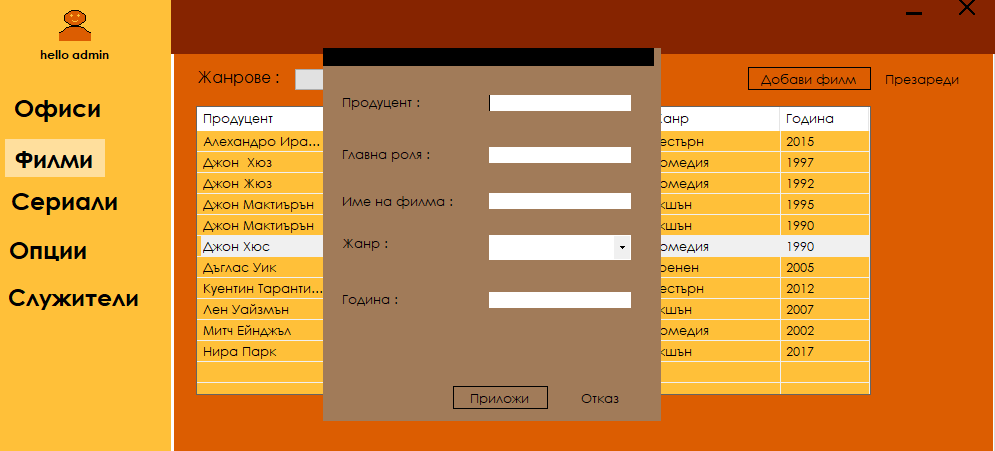
Начален екран на системата при вход във администраторски режим



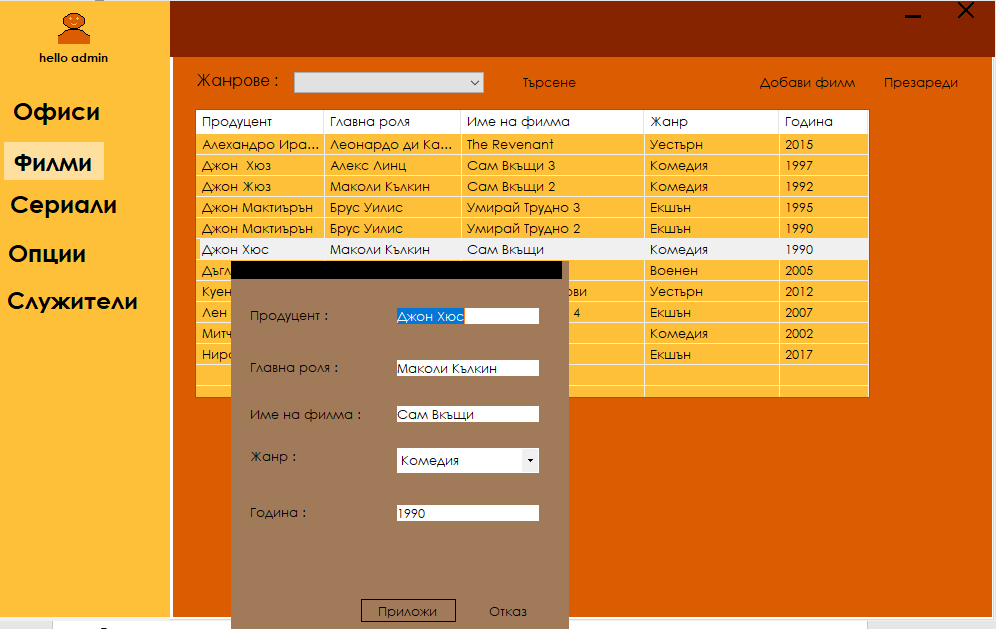
Екран, който позволява на администратора да добавя и коригира съществуващи филми



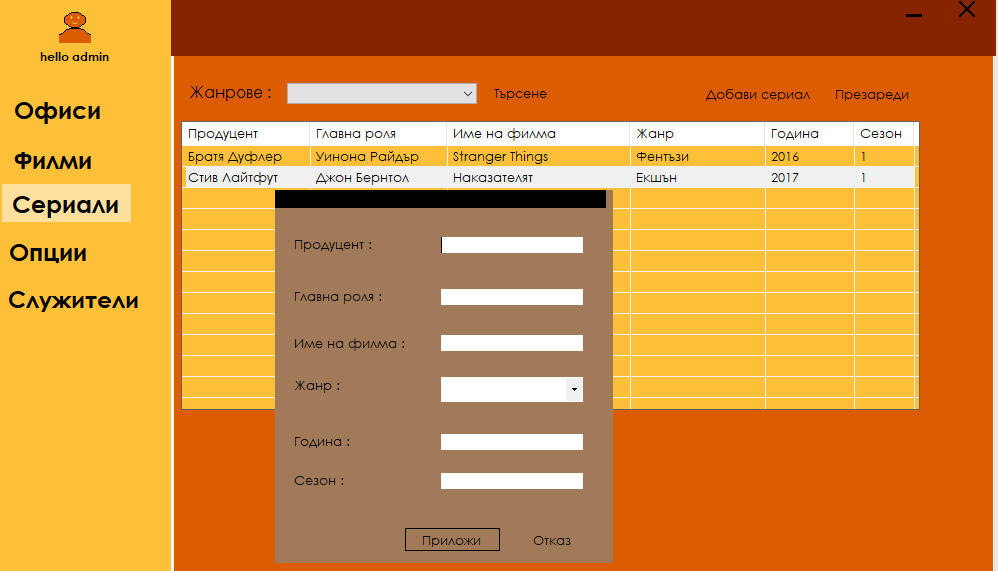
Екран при добавяне на филм



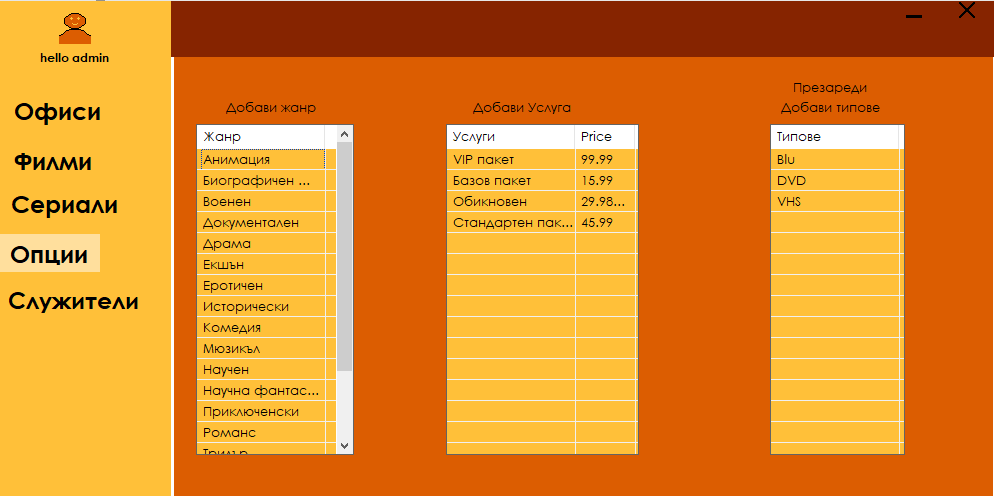
Екран при промяна на филм



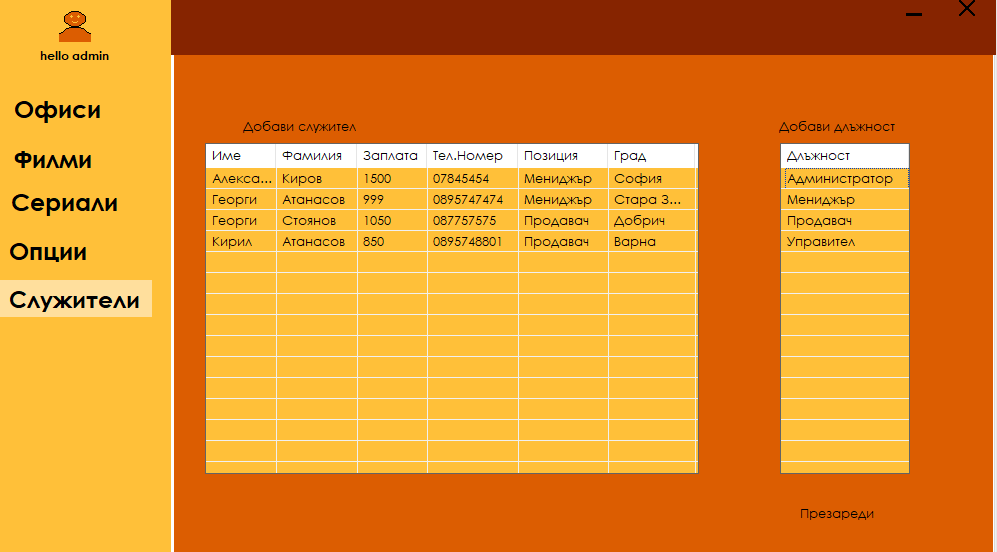
Екран при добавяне на сериал



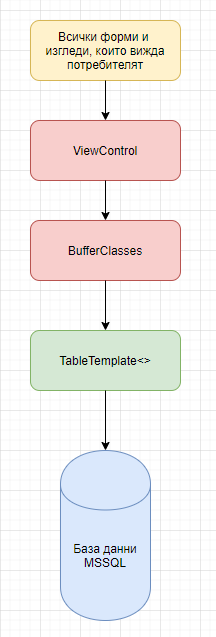
Екран, който позволява добавяне и промяна на жанрове, типовете услуги и типовете филми, който са в офисите



Екран, който позволява добавяне и промяна на служители и добавяне на длъжностти

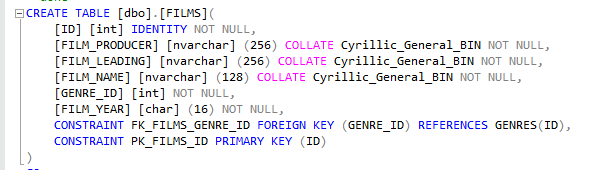


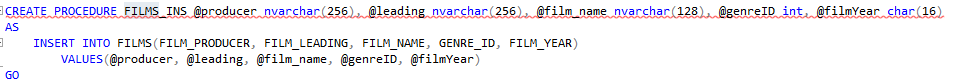
1. Програмни средства за реализация на системата
   1. Архитектура на системата

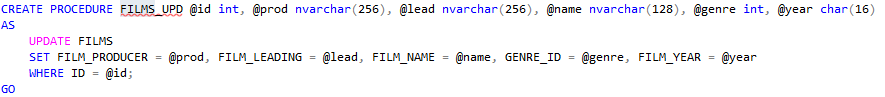
Архитектурата на системата има 5 слоя:

* Най долният слой е базата данни, която е реализирана чрез MSSQL, който е подобен на OracleSQL, но е създаден да обслужва по-големи системи
* След това е темплейтният клас, който комуникира със базата данни. Темплейтен е защото за всяка заявка към дадена таблица минава през този клас. Който според класа, който е подаден знае в коя таблица да търси данните. Примерно
* След това са така наречените буферни класове, който служат за съхранение на данните от таблиците от базата данни. Те ни помагат за по-голямата скорост, защото когато търсим нещо то се търси в масивам а не веднага в базата данни, което съществено забавя операцията
* След това класът ViewControl, които се намира по-голямата част от бизнес логиката. Използвам го за да зареждам определени данни в ListViewControl-ите за таблиците. Също така при този клас използвам Singleton design pattern-a, със цел да се работи винаги със по една инстанция на буферните класове. Във него зареждам определените данни спрямо това какъв е входа във системата потребителски или администраторски и така зареждам определени масиви
* И най-горният слой са всички форми и ListViewControl таблици, които използвам за изобразяване на „суровите данни“ на потребителя в по приемлив вид. В тези форми има изградени доста радио бутони и контекст менюта, който да направят работата на потребителя по лесна. Също така тук включвам класът за хеширане на данните на потребителя.
  1. Интересни фрагменти код и начини на създаване на елементите
     1. База данни – създаването на базата е обикновенно с таблици и връзките между тях. Като си улесних работата като създадох процедури за вкарване и промяна на данните за всяка таблица, за да ми е по лесно като извиквам заявките после от TableTemplate<> класа

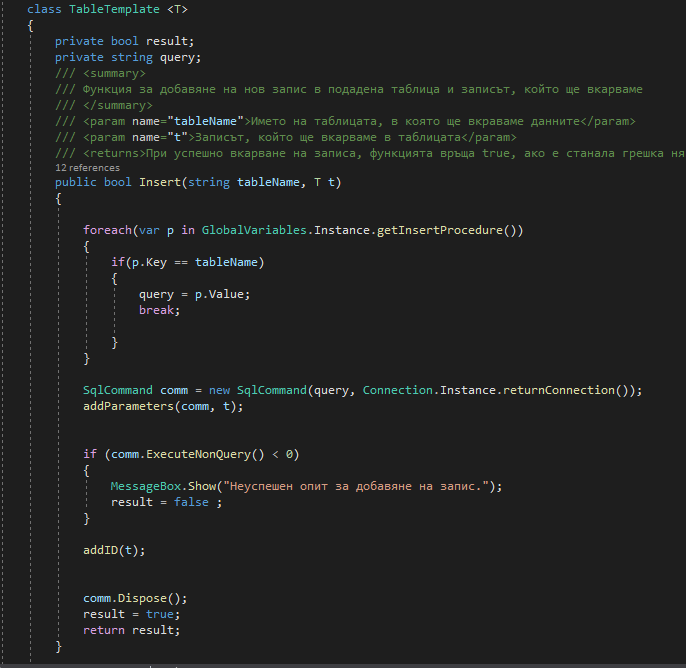
Например:

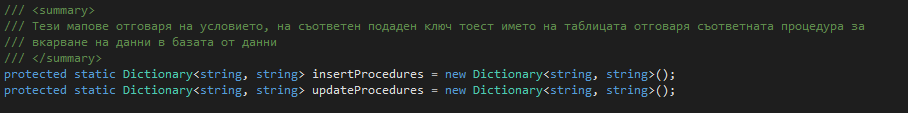
* Създаване на таблицата за съхранение на филми

 Създаване на процедура за вкарване на филм

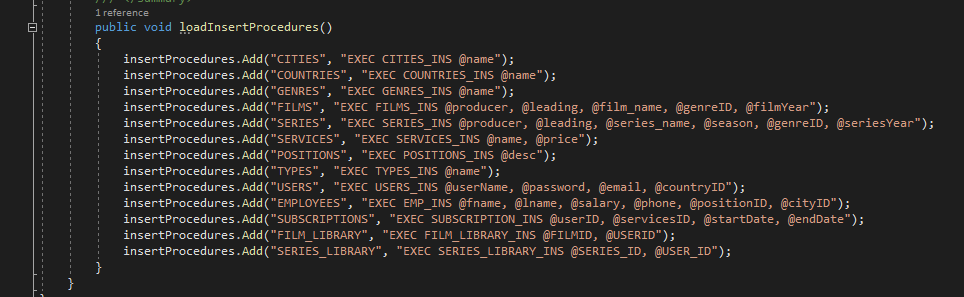
Създаване на процедура за промяна на филм

По този начин писането на заявките за добавяне и промяна е доста по-лесно.

* + 1. TableTemplate<> - създадох този клас със цел само той да комуникира с базата данни и за да няма повтарящи се класове. Тей като имам доста таблици и трябваше да имам доста класове, които имат еднакви функции, но просто със различни имена. Тей като класът е доста обемен ще сложа няколко фрагмента от код от този клас просто да го представя как работи.
* Тази функция е функцията за добавяне на запис в базата. Като подаваме името на таблицата, където ще добавяме и данните, които ще добавяме. Тук се набиватдвата класа **GlobalVariables** и **Connection**, при който също използвам Singleton design Pattern-a. Тези класове използвам за запазването на съответно глобалните променливи като процедурите и връзката към базата данни. При функцията **getInsertProcedure()** тук търся във един мап точната процедура за вкарване в базата данни за съответната таблица. На следващата снимка има пример със маповете. **ReturnConnection()** използвам да взема единствената връзка, която е създадена в началото на стартиране на системата.



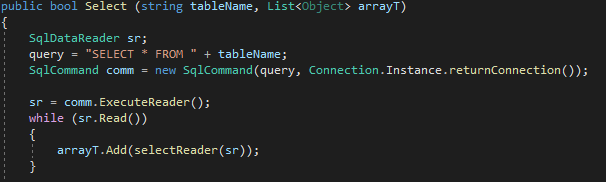
Ето примерите за съответствието на името на таблицата със нейната процедура за вкарване на данните. Същата работа е и при процедурите за промяна на данните.

Също така трябва да се добавят параметрите към тези процедури. С помоща на функцията **addParameters()**

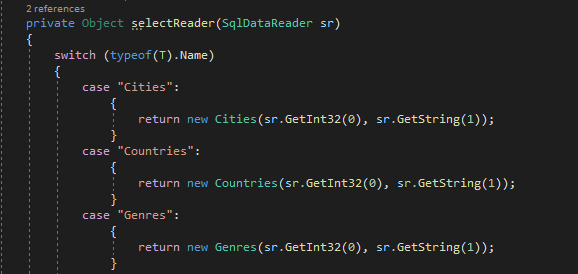
Където се попълват параметри-те във SqlCommand променли-вата

Същото важи и за параметри-те за промяна на данните.

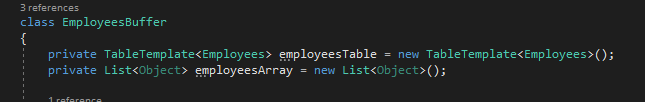
Интересен момент също се получава при извличането на данните от базата данни .



Където използвам функцията **selectReader()**, за да разгранича какъв клас ще чета от базата.



* + 1. BufferClasses – или буферни класове те са създадени за да запазват данните и да работят със тях по-бързо.

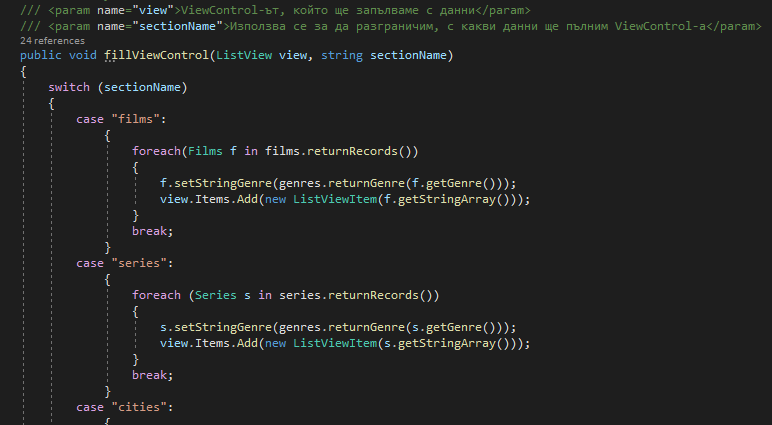
Това е буферният клас на служителите, който има **employeesTable** за да комуникира със базата данни и масивът от служители.

Това е примерна промяна на служител, като първо се проверява дали този служител съществува.

След това се намира служителя в масива и се променят неговите стойности и се изпраща към базата данни за да се запамети и там.

Тези буферни класове са доста удобни, защото улесняват доста целият работен процес по нагоре във архитектурата.

* + 1. ViewControl – този клас е най-сложният, тей като във него е включена цялата бизнес логика на приложението. И то със цел да има по една инстанция на буферните класове активни постоянно.



Този фрагмент е част от класът ViewControl, където функцията **fillViewControl()** попълва ListView таблицата по зададеният sectionName или разпределител, който ни показва с какви данни да попълним ListView таблицата.

Тази функция е част от ViewControl. Функцията връща и запълва таблицата спрямо резултата от извършеното търсене в масива.

Функцията запълва данните спрямо даденото sectionName.

public List<Object> searchByFilter(Films f) – тази е функцията, която търси и връща масивът, който изпълнява задедените условия

{

return getGenreSearch(f, filmsArray);

}

private List<Object> getGenreSearch(Films f, List<Object> array)

{

if (f.getGenre() != 0)

{

List<Object> resultArray = new List<Object>();

foreach (Films i in array)

{

if (i.getGenre() == f.getGenre())

{

resultArray.Add(i);

}

}

return this.getNameSearch(f, resultArray);

}

return this.getNameSearch(f, array);

}

private List<Object> getNameSearch(Films f, List<Object> array)

{

string name = f.getName();

if(!String.IsNullOrEmpty(name))

{

List<Object> resultArray = new List<Object>();

foreach (Films i in array)

{

if (i.getName() == f.getName())

{

resultArray.Add(i);

}

}

return getLeadSearch(f, resultArray);

}

return getLeadSearch(f, array);

}

private List<Object> getLeadSearch(Films f, List<Object> array)

{

if (!String.IsNullOrEmpty(f.getLeading()))

{

List<Object> resultArray = new List<Object>();

foreach (Films i in array)

{

if (i.getLeading() == f.getLeading())

{

resultArray.Add(i);

}

}

return getYearSearch(f, resultArray);

}

return getYearSearch(f, array);

}

private List<Object> getYearSearch(Films f, List<Object> array)

{

if(f.getYear() != 0)

{

List<Object> resultArray = new List<Object>();

foreach (Films i in array)

{

if (i.getYear() == f.getYear())

{

resultArray.Add(i);

}

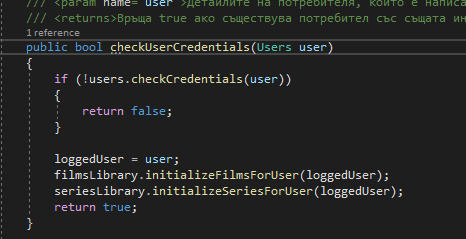
}

return resultArray;

}

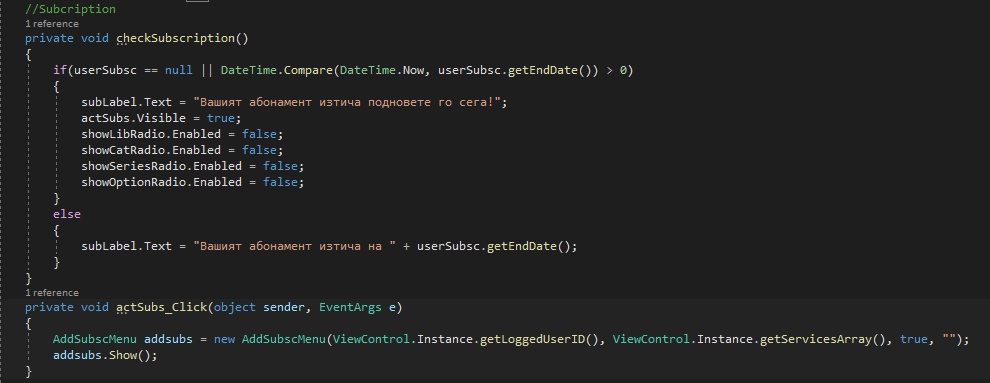
return array;

}

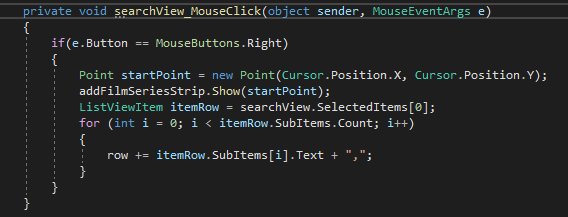
Тази функция е част от ViewControl и я използвам за да проверя данните на потребителя. След като намеря съответствие на този потребител в масива от потребители след това го запазвам в променливата **loggedUser** и след това пълня масивите за любимите филми и сериали за този потребител за да мога после да визуализирам данните.

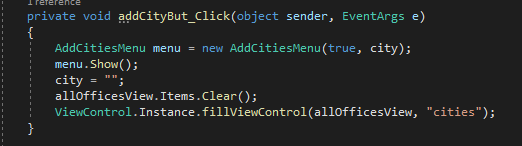
* + 1. Формите, и всички начини на представяне на информацията на потребителя – за главно представяне на данните реших да представям данните със ListVIew, защото изглежда доста добре и има възможност за маркиране на един елемент, което ми позволява да взема един елемент за да мога да го променя.

Главното меню, което използвам за работата на потребителя е MainMenu, което се съдържа няколко елементи. Използвам **TabControl-и**  за да мога да превключвам на различните сцени, които са свързани със радио бутоните. Имам един TabControl за потребителският вход и един за администраторският вход. И спрямо какъв е входът във системата, определям кой TabControl да се зареди.

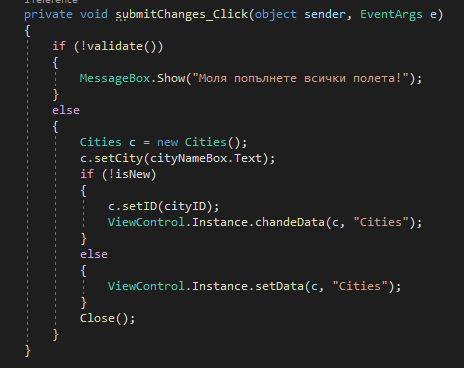


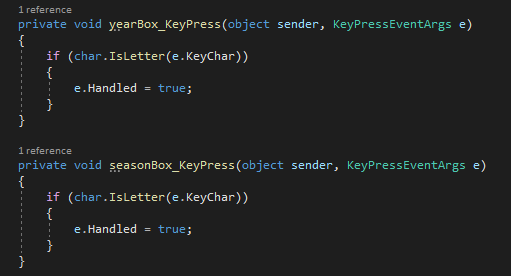
Например тази функция е част от началното меню на потребителският вход и тук проверявам дали влезлият потребител има активен абонамент. Ако няма такъв се блокират всички радио бутони и той не може да използва системата докато не закупи даденият абонамент.

Тази функция създава контекстно меню, което го използвам за да добавя филм или сериал в хранилището на потребителя. Като взимам избраният елемент и го запаметявам временно докато се цъкне бутона за вкарване на филма или сериала от контекстното меню и след това го изтривам.

Тази функция я използвам за добавяне на нов град. Като създавам форма, която използвам за да въведа името на града .

За всяко въвеждане имам създадена форма, която също така има създадена валидация, която проверява интегритета на данните, които ще се вкарват в базата данни.

Тази функция е част от формата за съдаване на градовете. Ако функцията validate() върне false то няма да бъде вкарана или променена базата данни. За да разгранича дали ще вкарвам в базата или ще променям аз използвам този флаг, който ако е false означава че този елемент ще се променя при натискането на submit бутона ако е true то тогава ще се добавят данните в базата данни.

Този фрагмент е от формата за добавяне на филми или сериали. Където в полетата за година и сезон, задължително трябва да се въвеждат само числа. Затова използвам функцията KeyPress и проверявам натиснатият бутон в момента на записването и ако е символ, а не число той просто не се изписва.

1. Развитие на продукта
   1. Създаване на клиент-сървър, които да обработва заявките към базата данни със цел да може да се отваря системата на повече клиенти
   2. Да се създаде модул за заплащане онлайн, като да може да се извършва онлайн транзакции и плащане с карта
   3. Модул за гледане на филмите онлайн.
2. Използвана литература.
   1. Литература за WindowsForms C# - <https://docs.microsoft.com/en-us/dotnet/framework/winforms/>
   2. Литература за MSSQL - https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15