**Оглавление**

[Аннотация 4](#_Toc26196310)

[Введение 5](#_Toc26196311)

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ 6](#_Toc26196312)

[1.1 Основное функциональное назначение программного средства 6](#_Toc26196313)

[1.2 Полное наименование программного средства 6](#_Toc26196314)

[1.3 Условное обозначение программного средства 6](#_Toc26196315)

[1.4 Разработчики программного средства 6](#_Toc26196316)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc26196317)

[2.1 Основание для разработки 7](#_Toc26196318)

[2.2 Назначение разработки 7](#_Toc26196319)

[2.3 Требование к программному средству 7](#_Toc26196320)

[2.4 Требования к программной документации 8](#_Toc26196321)

[2.5 Требования к эргономике и технической эстетике 8](#_Toc26196322)

[2.6 Стадии и этапы разработки 9](#_Toc26196323)

[2.7 Порядок контроля и приемки 10](#_Toc26196324)

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 11](#_Toc26196325)

[3.1 Декомпозиция поставленной задачи 11](#_Toc26196326)

[3.2 Общая архитектура программного средства 12](#_Toc26196327)

[3.3 Разработка алгоритма решения задачи 15](#_Toc26196328)

[3.4 Реализация функционального назначения программного средства 16](#_Toc26196329)

[3.5 Структурная организация данных 16](#_Toc26196330)

[3.6 Разработка интерфейса ПС 27](#_Toc26196331)

[3.7 Описание структуры выходной информации 3](#_Toc26196332)2

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 3](#_Toc26196333)3

[4.1 Назначение программного средства 3](#_Toc26196334)3

[4.2 Условия выполнения программного средства 3](#_Toc26196335)3

[4.3 Эксплуатация программного средства 3](#_Toc26196336)3

[4.4 Сообщения пользователю 4](#_Toc26196337)0

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 4](#_Toc26196338)2

[5.1 Обоснование необходимого количества тестов 4](#_Toc26196339)2

[5.2 Описание тестовых пакетов 4](#_Toc26196340)3

[6. ЗАКЛЮЧЕНИЕ 4](#_Toc26196341)9

[6.1 Оценка качества программного средства с помощью метрик 4](#_Toc26196342)9

[7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 5](#_Toc26196343)2

[Приложение А листинг программного модуля 5](#_Toc26196344)4

[Приложение B диск с исполняемым кодом программного модуля 7](#_Toc26196345)7

**Аннотация**

На курсовой проект студента Халилуллина К.И. на тему: Разработка интегрированного программного модуля «Театр». Курсовой проект представлен пояснительной запиской на 65 листов машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве Театр. В главе 2 приведено техническое задание на проектирование БД согласно ГОСТ 19.201. Глава 3 содержит пояснительную записку к программному продукту согласно ГОСТ 19.404-79. В главе 4 содержится руководство пользователя согласно ГОСТ 19.504-79. Глава 5 содержит обоснование количества необходимых тестов и тестовые пакеты для оценки работоспособности программного продукта. В главе 6 приведены результаты ручного и автоматического тестирования программного продукта и выполнена оценка качества с помощью метрик.

В работе содержится общее описание программного средства, предназначенного для объекта проектирования – интегрированного программного модуля «Театр», изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**Введение**

В курсовом проекте была поставлена задача разработки интегрированного программного модуля «Театр», который позволяет автоматизировать учет спектаклей.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов был выбран язык программирования Java. Выбор обусловлен тем, что возможностей этого языка достаточно для достижения поставленной цели моей курсового проекта.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа, а также тип пользователей, которые с ней будут работать.

**1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для автоматизации учета спектаклей. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

Разработка интегрированного программного модуля «Театр»

**1.3 Условное обозначение программного средства**

«TheaterAp»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 215 Халилуллин К.И.

**2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

Работа ведётся на основании задания на курсовой проект по специальности 09.02.07 — «Информационные системы и программирование».

**2.2 Назначение разработки**

Программа «TheaterAp» кросс-платформенная. Программа предназначена для автоматизации деятельности администратора по учету спектаклей театра. Программа позволяет вводить, изменять и удалять данные спектаклей, просматривать информацию.

**2.3 Требование к программному средству**

«TheaterAp» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 4 Гб;

Процессор: Intel Core i / Phenom II X4 / AMD Athlon64 X2 4600 + 2.4 ГГц;

Видео карта: nVidia GeForce 8600 / ATI Radeon HD 2400;

Звуковая карта: Совместимая с DirectX;

Программный продукт «TheaterAp» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«TheaterAp» написана на объектно-ориентированном языке программировании Java в интегрированной среде разработки IntelliJ IDEA. В качестве СУБД используется PostgreSQL. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1. техническое задание;

2. сроки выполнения работ;

3. программу и методику испытаний;

4. эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя;

- взаимодействие пользователя с системой осуществляется на двух языках по выбору (русский и английский), для наибольшего удобства

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;

2. рабочее проектирование;

3. внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на нее;

5. выбор языков программирования;

6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. разработка, согласование и утверждение порядка и методики испытаний;

2. проведение испытаний;

3. корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ**

**3.1 Декомпозиция поставленной задачи**

Для достижения цели работы была поставлена следующая задача:

Создать модуль “CodeDatabase” отвечающий за сохранение БД. Результат работы модуля - сохранение БД.

Поставленные задачи наглядно отображаются на диаграмме деятельности.



Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является спектакль.

Рассматриваемые процессы: Добавление и удаление информации о спектаклях.

Объекты моделирования представлены на диаграмме классов, рис.2.

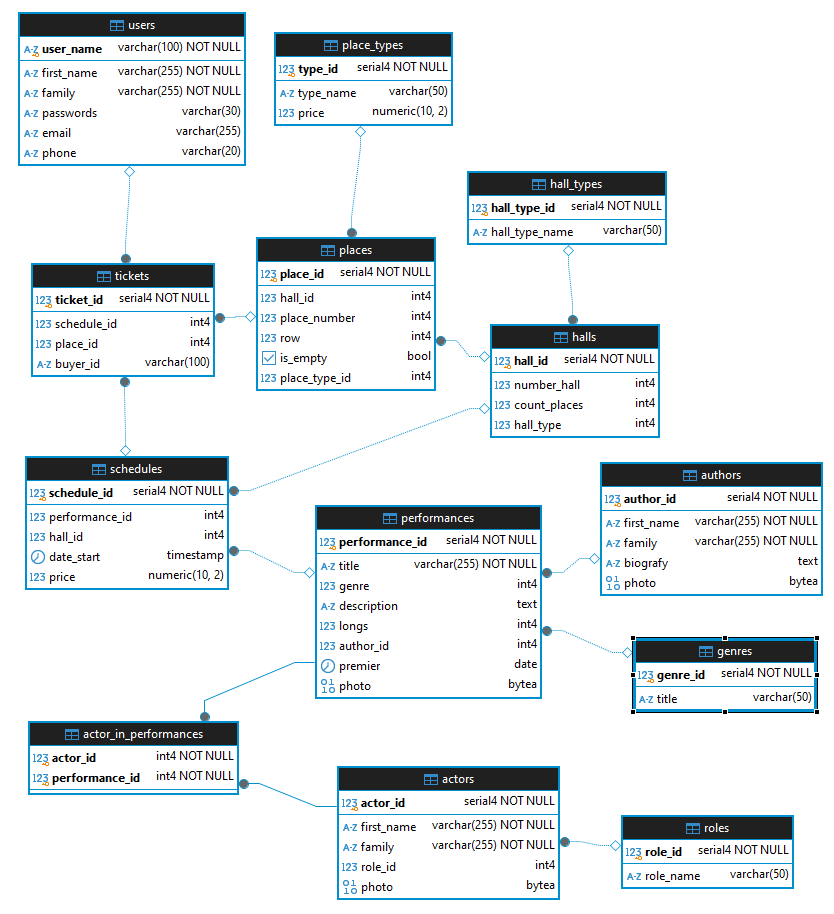


Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.



Рис.3 Схема событийно-ориентированной архитектуры

В «TheaterAp» используется событийно-ориентированная архитектура.(Рис.3) Роль агента (источник событий) в программе выполняют: пользователи программы, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие: добавить, удалить, редактировать, и т.д. система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.

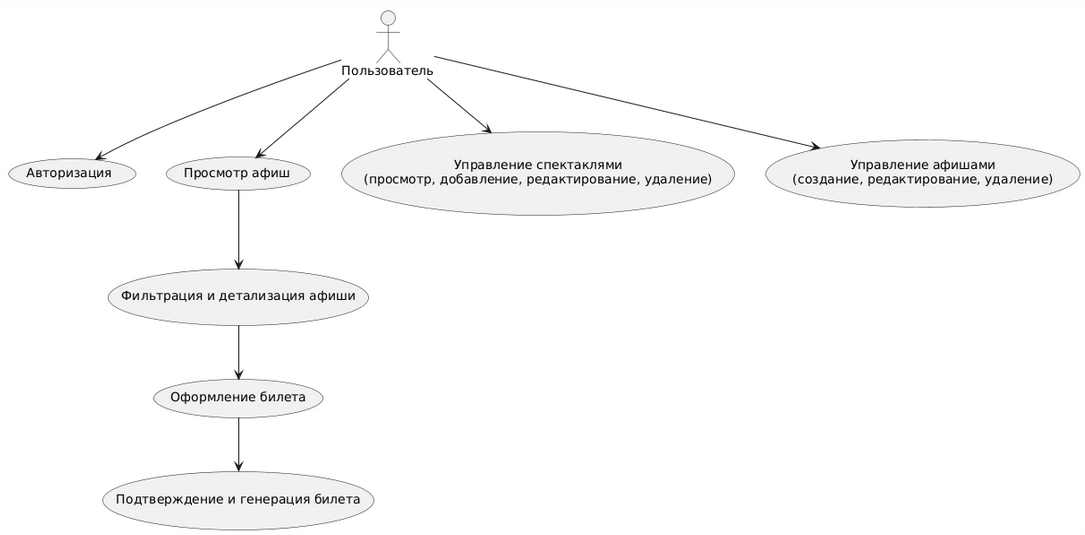


Рис.4 Диаграмма вариантов использования

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

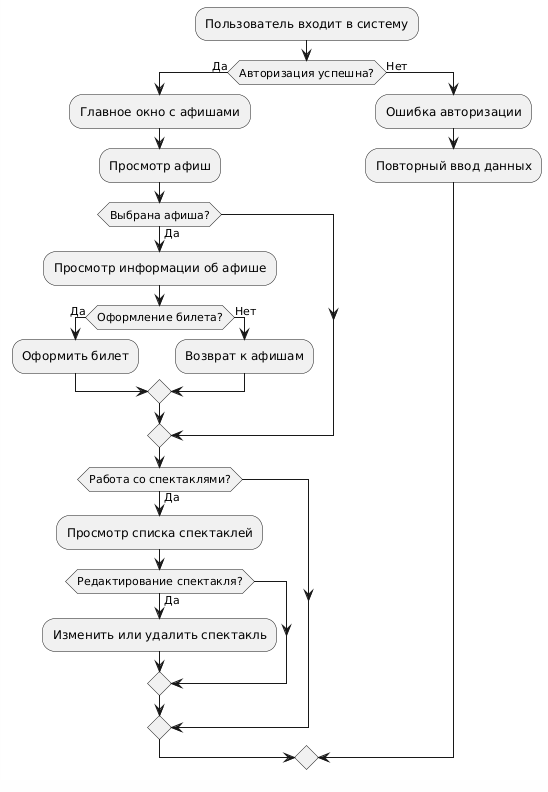


Рис.5 Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «TheaterAp» имеет следующий набор входных данных, такие как: спектакли и информация о них.

Данные вводятся администратором в соответствующие поля ввода, снабженные всплывающими подсказками.

Выходными данными являются: отображение списка мероприятий в виде плиточного интерфейса.

Данные вводятся пользователем в соответствующие поля вывода, снабженные всплывающими подсказками.

Выходные данные редактировать вручную невозможно, т.к. они служат в качестве отчетной информации для пользователя.

**3.5 Структурная организация данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы. Все эти данные указаны в реляционной модели «БД Театр» представленная на рисунке. Любая реляционная база данных и называется реляционной, что характеризуется отношениями (relation) между таблицами. На рисунке изображены таблицы моей базы данных. При этом одна таблица является родительской (главной), а вторая – дочерней (подчиненной). Главной таблицей является «Представление». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

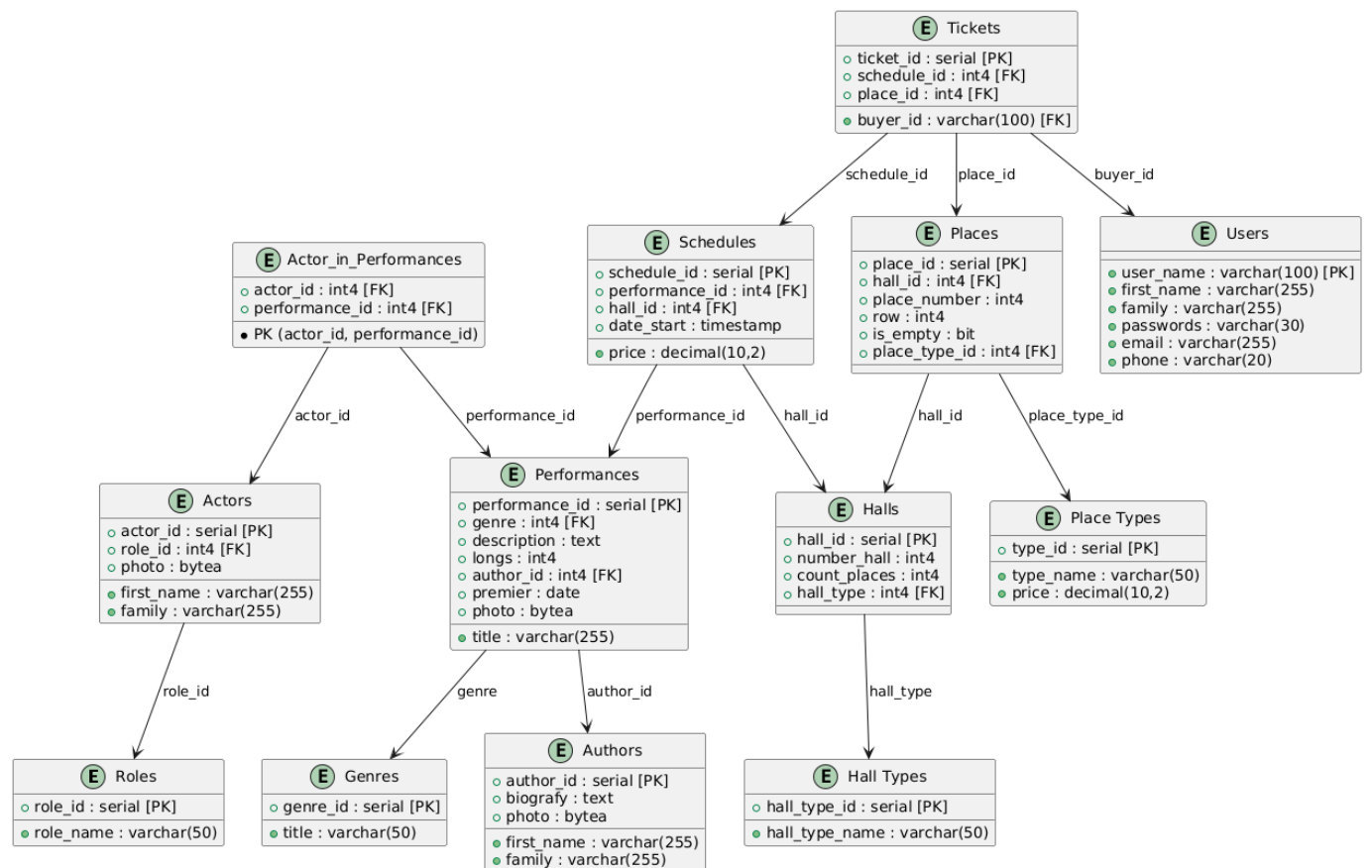


Рис.6 Реляционная модель

Первичный ключ в базе уникален, используется для организации отношений между таблицами, который не может иметь пустых и повторяющихся значений. Первичными ключами в базе являются поля: Имя\_пользователя (таблица Пользователи), Код\_билета (Таблица Билеты), Код\_расписания (Таблица Расписание), Код\_места (Таблица Расписание), Код\_типа\_места (Таблица Тип места), Код\_зала (Таблица Залы), Код\_представления (Таблица Представления), Код\_типа\_зала (Таблица Типы зала), Код\_жанра (Таблица Жанры), Код\_автора (Таблица Авторы), Код\_актера (Таблица Актеры), Код\_роли (Таблица Роли). Остальные ID-ключи являются внешними ключами.

Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:

* БД находится в форме - 1НФ потому, что
  + Таблица не имеет повторяющихся записей;
  + Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
  + Таблица не имеет повторяющихся групп полей.
    - Вторая нормальная форма(2НФ):
      * Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора, т.е. ID.
        + Третья нормальная форма(3НФ):

Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

На основе реляционной модели базы данных мною в PostgreSQL была построена следующая база данных. (Рис.7) В ней первичным ключом является поле таблицы “performances” – “performance\_id”. Связь таблиц “genres” - “performances” по полю “genre\_id” и “authors” - “performances” по полю “author\_id” один ко многим (1:М). Таблицы “hall\_types” и “halls” связаны по полю “hall\_type” типом связи 1:М. Таблицы “places” и “halls” связаны по полю “hall\_id” , а так же “place\_types” и “places” связаны по полю “place\_type\_id” типом связи 1:М. Таблицы “roles” и “actors” связаны по полю “role\_id” типом связи 1:М. Таблицы “actor” и “actor\_in\_performances” связаны по полю “actor\_id” типом связи 1:М, а так же таблицы “performances” и “actor\_in\_performances” по полю “perfomance\_id” типом связи 1:М. Таблица “schedules” связана с таблицами “performances” по полю “performance\_id” и “halls” по полю “hall\_id” типом связи 1:М. Таблица “tickets” связана с таблицами “places” по полю “place\_id”, таблицей “schedules” по полю “schedule\_id” и таблицей “users” по полю “user\_id” типом связи 1:М. Все связи между таблицами базы данных типа один ко многим.

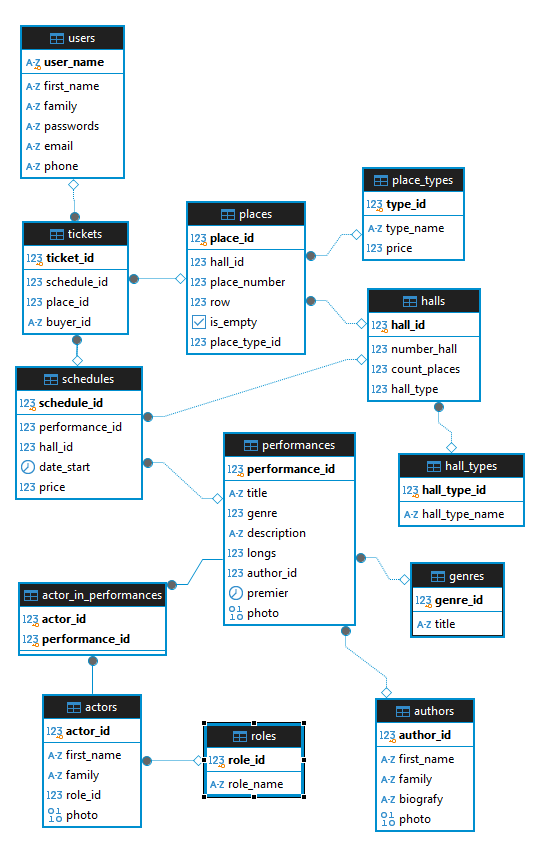


Рис.7 Схема базы данных

Таблица “actor\_in\_performances”(Актер в спектакле)

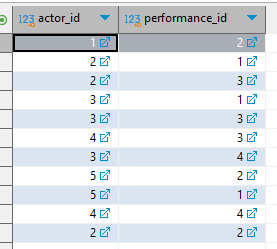


Рис.8 Таблица actor\_in\_performances

Таблица “actors”(актеры)



Рис.9 Таблица Category

Таблица “authors”(Авторы)

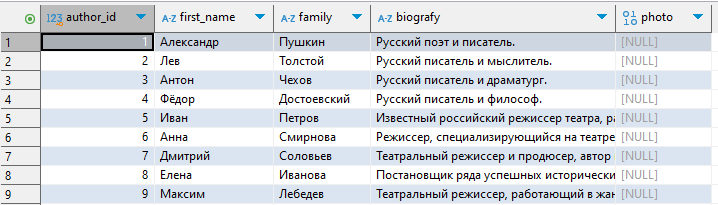


Рис.10 Таблица authors

Таблица “genres”(Жанры)

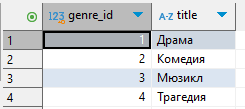


Рис.11 Таблица genres

Таблица “hall\_types”(Типы зала)

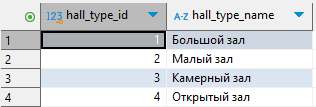


Рис.12 Таблица hall\_types

Таблица “halls”(Залы)

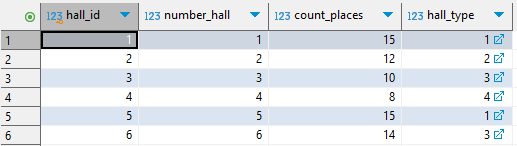


Рис.13 Таблица halls

Таблица “performances”(Спектакли)

Рис.14 Таблица performances

Таблица “place\_types”(Типы места)

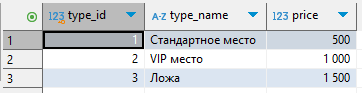


Рис.15 Таблица place\_types

Таблица “places”(Места)

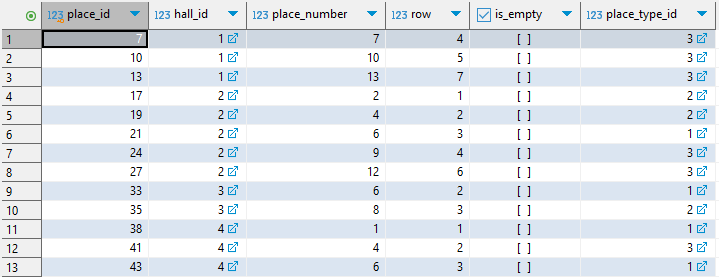


Рис.16 Таблица places

Таблица “roles”(Роли)

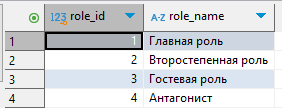


Рис.17 Таблица roles

Таблица “schedules”(Расписание)

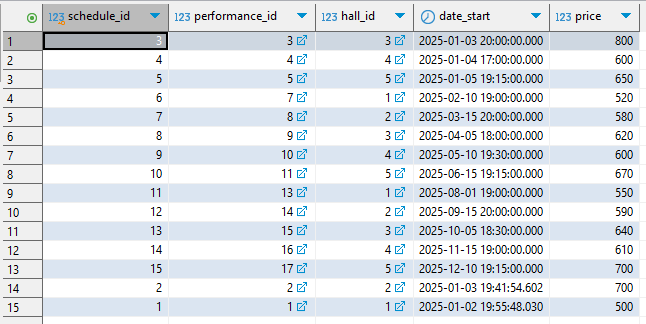


Рис.18 Таблица schedules

Таблица “tickets”(Билеты)

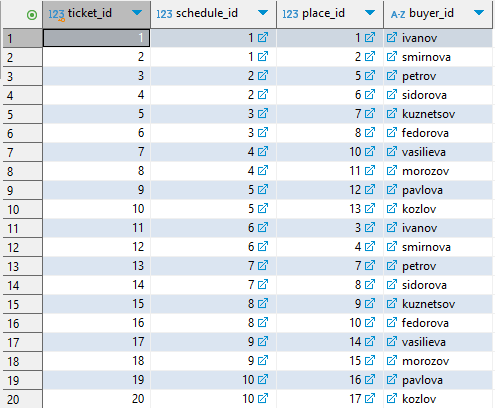


Рис.19 Таблица tickets

Таблица “users”(Пользователи)

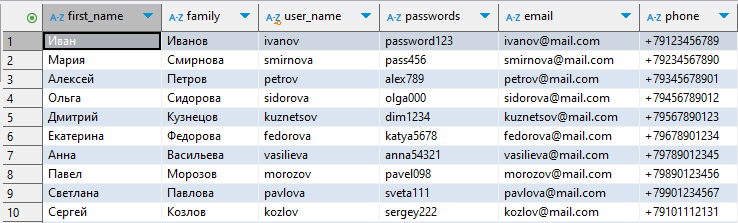


Рис.20 Таблица users

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Словарь данных | | | | | | | | |
|  | | | | | | | | |
| hall\_types | | | | | | | | |
| key | title | | fieldType | | required? | | note | |
| PK | hall\_type\_id | | serial4 | | Y | |  | |
|  | hall\_type\_name | | varchar(50) | |  | |  | |
|  |  | |  | |  | |  | |
| halls | | | | | | | | |
| key | | title | | fieldType | | required? | | note |
| PK | | hall\_id | | serial4 | | Y | |  |
| FK | | hall\_type | | int4 | |  | |  |
|  | | number | | int4 | |  | |  |
|  | | count\_place | | int4 | |  | |  |
|  | |  | |  | |  | |  |
| performances | | | | | | | | |
| key | | title | | fieldType | | required? | | note |
| PK | | spectakle\_id | | serial4 | | Y | |  |
|  | | title | | varchar(255) | |  | |  |
|  | | genre | | int4 | |  | |  |
|  | | description | | text | |  | |  |
| FK | | author\_id | | int4 | |  | |  |
|  | | premier | | date | |  | |  |
|  | | photo | | bytea | |  | |  |
|  | | long | | int4 | |  | |  |
|  | |  | |  | |  | |  |
| actors | | | | | | | | |
| key | | title | | fieldType | | required? | | note |
| PK | | actor\_id | | serial4 | | Y | |  |
|  | | first\_name | | varchar(255) | | Y | |  |
|  | | family | | varchar(255) | | Y | |  |
| FK | | role\_id | | int4 | |  | | ссылка на roles |
|  | | photo | | bytea | |  | |  |
|  | |  | |  | |  | |  |
| places | | | | | | | | |
| key | | title | | fieldType | | required? | | note |
| PK | | place\_id | | serial4 | | Y | |  |
|  | | ryad | | int4 | |  | |  |
|  | | number | | int4 | |  | |  |
| FK | | hall\_id | | int4 | |  | |  |
| FK | | place\_type\_id | | int4 | |  | |  |
|  | | is\_empty | | boolean | |  | |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  | |  | |
| schedules | | | | | | | |
| key | title | fieldType | | required? | | note | |
| PK | schedule\_id | serial4 | | Y | |  | |
|  | date\_start | timestamp | |  | |  | |
| FK | performance\_id | int4 | |  | |  | |
|  | hall\_id | int4 | |  | |  | |
|  | price | numeric(10,2) | |  | |  | |
|  |  |  | |  | |  | |
| place\_type | | | | | | | |
| key | title | fieldType | | required? | | note | |
| PK | place\_type\_id | serial4 | | Y | |  | |
|  | place\_type\_name | varchar(50) | |  | |  | |
|  | price | numeric(10,2) | |  | |  | |
|  |  |  | |  | |  | |
| tickets | | | | | | | |
| key | title | fieldType | | required? | | note | |
| PK | tickets\_id | serial4 | | Y | |  | |
| FK | schedule\_id | int4 | |  | |  | |
| FK | place\_id | int4 | |  | |  | |
| FK | buyer\_id | int4 | |  | |  | |
|  |  |  | |  | |  | |
| users | | | | | | | |
| key | title | fieldType | | required? | | note | |
| PK | user\_name | varchar(100) | | Y | |  | |
|  | first\_name | varchar(255) | | Y | |  | |
|  | family | varchar(255) | | Y | |  | |
|  | email | varchar(255) | |  | |  | |
|  | phone | varchar(20) | |  | |  | |
|  | password | varchar(30) | |  | |  | |
|  |  |  | |  | |  | |
| authors | | | | | | | |
| key | title | fieldType | | required? | | note | |
| PK | author\_id | serial4 | | Y | |  | |
|  | first\_name | varchar(255) | | Y | |  | |
|  | family | varchar(255) | | Y | |  | |
|  | biografy | text | |  | |  | |
|  | photo | bytea | |  | |  | |
|  |  |  | |  | |  | |
| actor\_in\_performances | | | | | | | |
| key | title | fieldType | | required? | | note | |
| FK | actor\_id | int4 | | Y | |  | |
| FK | peerformance\_id | int4 | | Y | |  | |
|  |  |  | |  | |  | |
|  |  |  | |  | |  | |
| genres | | | | | | | |
| key | title | fieldType | | required? | | note | |
| PK | genre\_id | serial4 | | Y | |  | |
|  | title | varchar(50) | |  | |  | |
|  |  |  | |  | |  | |
| roles | | | | | | |
| key | title | fieldType | required? | | note | |
| PK | role\_id | serial4 | Y | |  | |
|  | role\_name | varchar(50) |  | |  | |

**3.6 Разработка интерфейса ПС**

В приложении предусмотрена авторизация. Для запуска приложения необходимо щелкнуть дважды левой кнопкой мыши по ярлыку.

После старта приложения пользователь вводит свои логин и пароль , а далее приступает к работе с главной формой программы, на которой и будут отображаться все спектакли, которые есть в базе данных в таблице “расписания”. «TheaterAp» обладает удобным интерфейсом, так что даже неопытному в обращение с пользовательскими компьютерными программами пользователю не составит труда быстро привыкнуть к работе с данным программным средством.

На рабочем экране расположены несколько кнопок для комфортной работы с приложением.

**Форма Авторизации.** Форма приложения, которую встречает пользователь и вводит свои логин, пароль для доступу к основному интерфейсу приложения.(Рис.21)

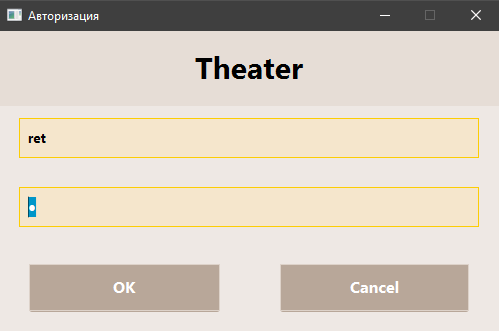
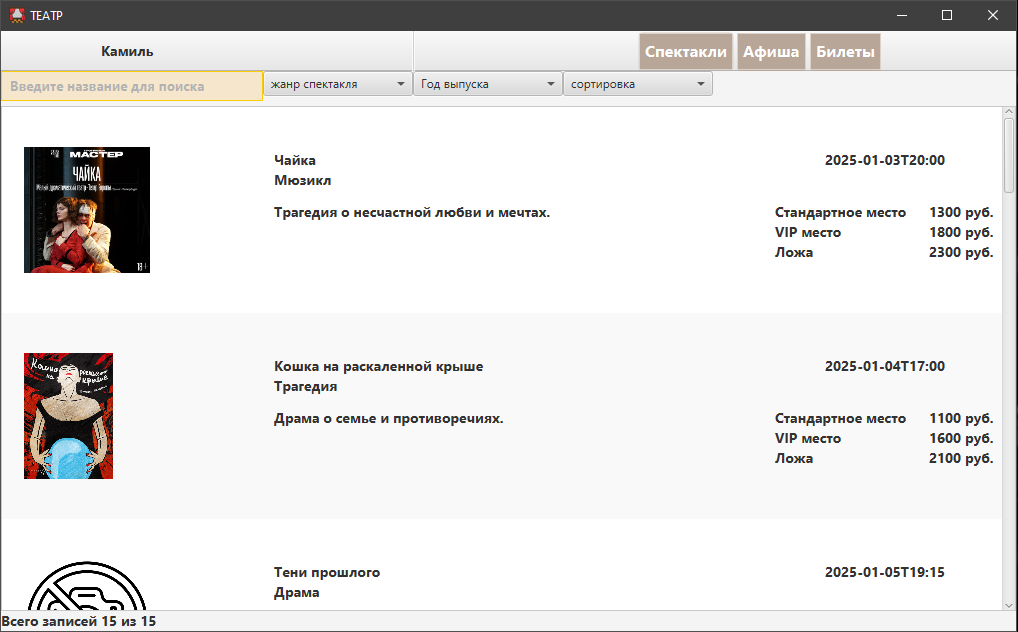


Рис.21 Авторизация

**Форма Расписания**. Основная форма приложения, в которой в виде плитки отображаются афиши спектаклей. Компоненты формы: ListView – для отображения расписания, Combobox – выпадающие списки для сортировки, TextBox и Button и строка поиска - TextField. (Рис.22)



TextField

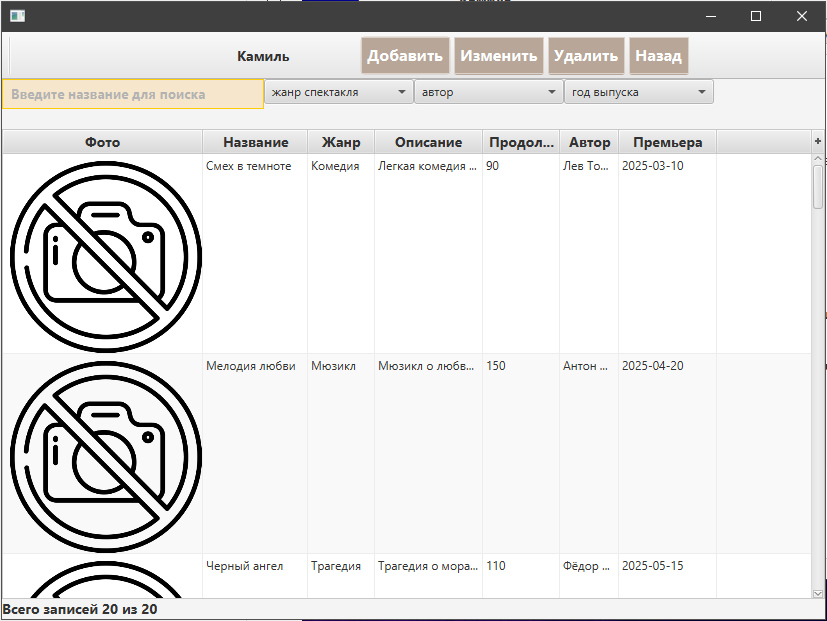
Button

ComboBox

TableView

Рис.22 Расписание

**Страница Спектакли**. Компоненты Страницы: TableView – для отображения списка спектаклей и кнопки Button для редактирования, добавления, удаления информации о спектаклях и переход на предыдущую страницу, а так же TextField ComboBox для фильтрации записей. (Рис.23)

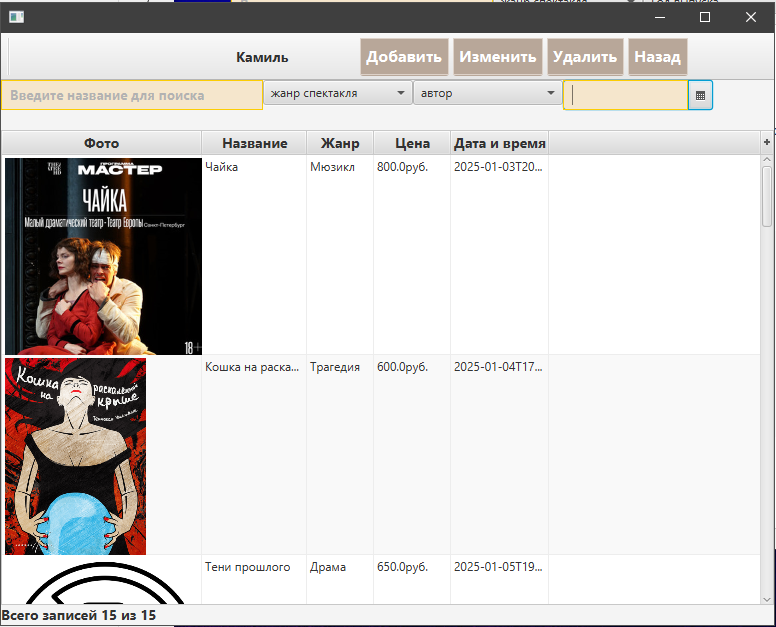


Button

TextField

Рис.23 Спектакли

**Страница Афиша**. Компоненты Страницы: TableView – для отображения списка афиш и кнопки Button для редактирования, добавления, удаления информации об афишах и переход на предыдущую страницу, а так же TextField, ComboBox, DatePicker для фильтрации записей. (Рис.24)



DatePicker

Button

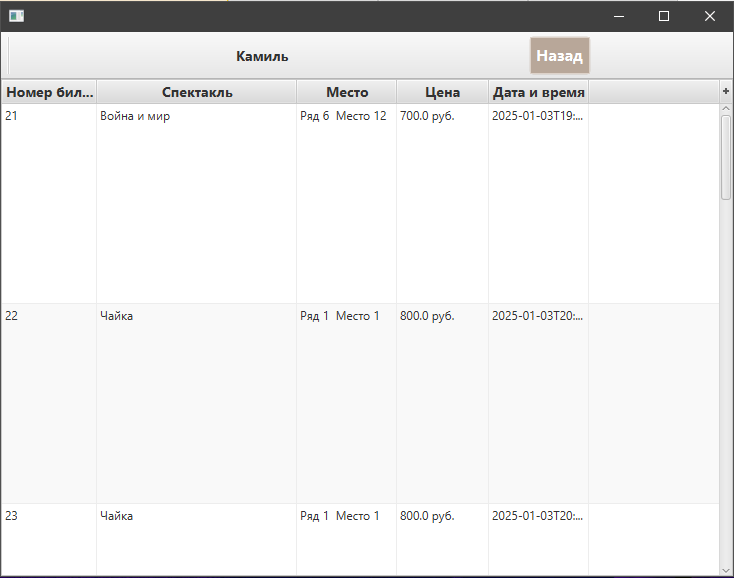
TableView

TextField

Button

Рис.24 Афиша

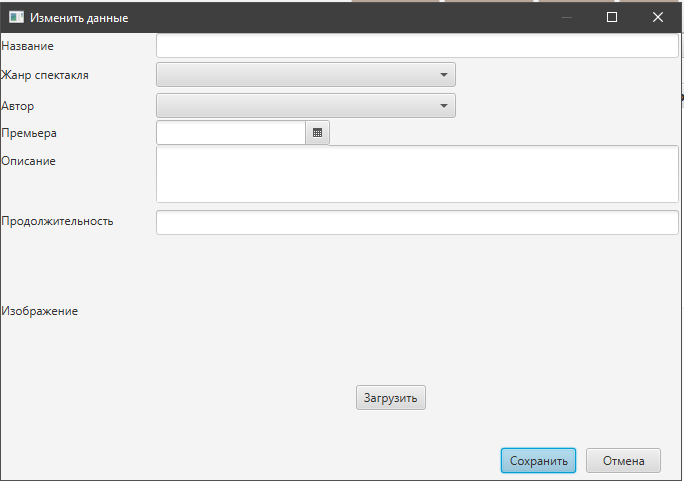
**Страница Билеты**. Компоненты Страницы: TableView – для отображения списка билетов пользователя и кнопки Button для перехода на предыдущую страницу. (Рис.25)



TableView

Рис.25 Билеты

**Страница Добавление и редактирование**. Интерфейс данной формы состоит из компонентов: TextBox – поля ввода, ComboBox – выпадающие списки, ImageView – для отображения изображения и Button. Данная страница открывается с помощью кнопок “Добавить” “Изменить” на страницах Спектакли, Афиша, Билеты. (Рис.26, Рис.27)



Imageview

Кнопки Сохранить Отмена

Рис.26 Страница Добавление и редактирование Спектакля

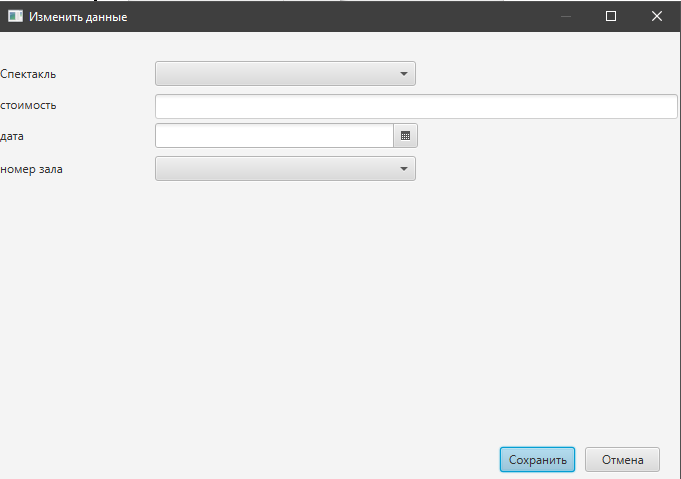
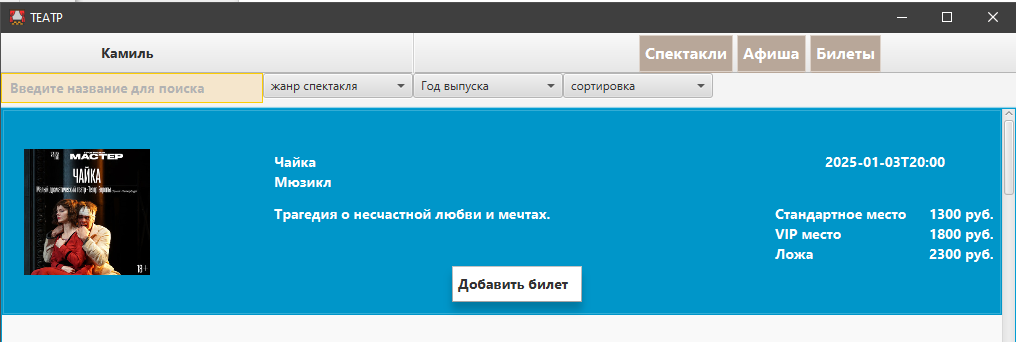


Рис.27 Страница Добавление и редактирование Афиши

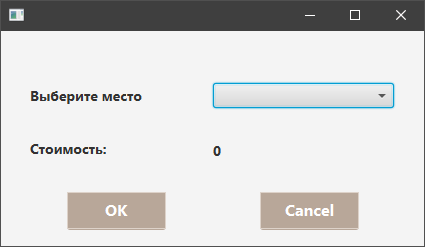
**Страница оформления билета**. Интерфейс данной формы состоит из компонентов: ComboBox и кнопок для его открытия необходимо нажать правой кнопкой мыши на любой из Спектаклей на форме расписания, а затем на появившуюся кнопку Добавить билет, предварительно выбрав место.(Рис.28-29)



Кнопка добавить билет

Кнопки Ок и Отмена

Рис.28 Открытие окна



Список мест для выбора

Вывод стоимости билета

Рис.29 Окно оформления билета

**3.7 Описание структуры выходной информации**

Выходными данными являются списки афиш отфильтрованные по разным критериям.

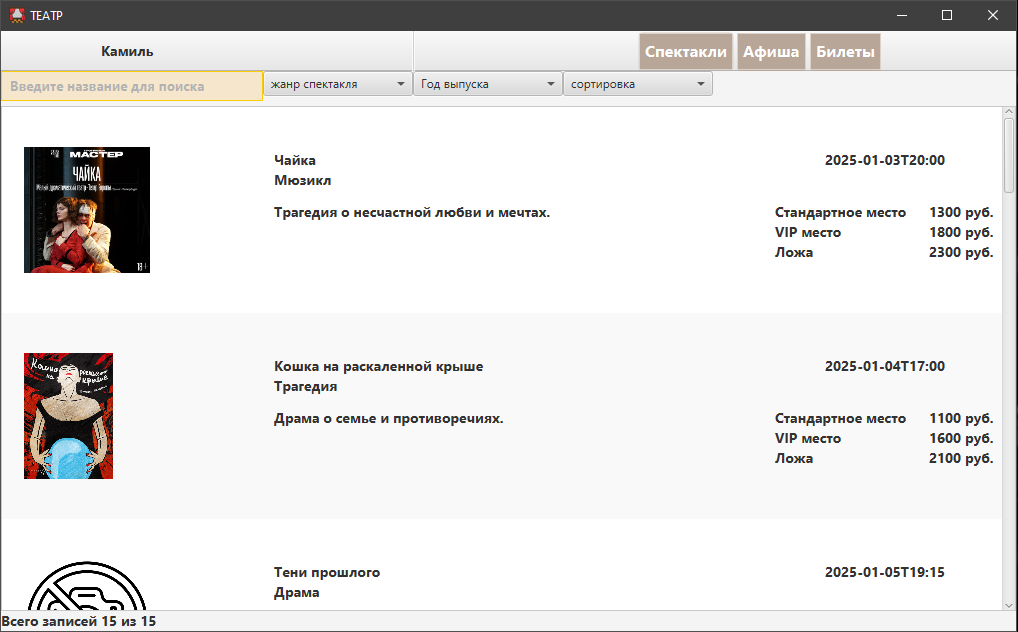


Рис.30 Списки мероприятий

**4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация учета спектаклей.

**4.2 Условия выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

-Компьютер с установленной на нём «TheaterAp».

Подготовительные действия:

-На иконке «TheaterAp» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «TheaterAp» и ее основных параметров.

***Активация продукта***

Активировать копию программы TheaterAp не нужно, т.к она бесплатна и находится в открытом доступе.

Вводные сведения об интерфейсе пользователя

После старта приложения откроется главное окно приложения – Каталог афиш в виде списка. На форме есть кнопка для перехода на страницу мероприятий. Есть поля для поиска афиши по назвванию и выпадающие списки для фильтрации по году выпуска спектакля, его жанру и сортировка по возрастанию. (Рис.31)

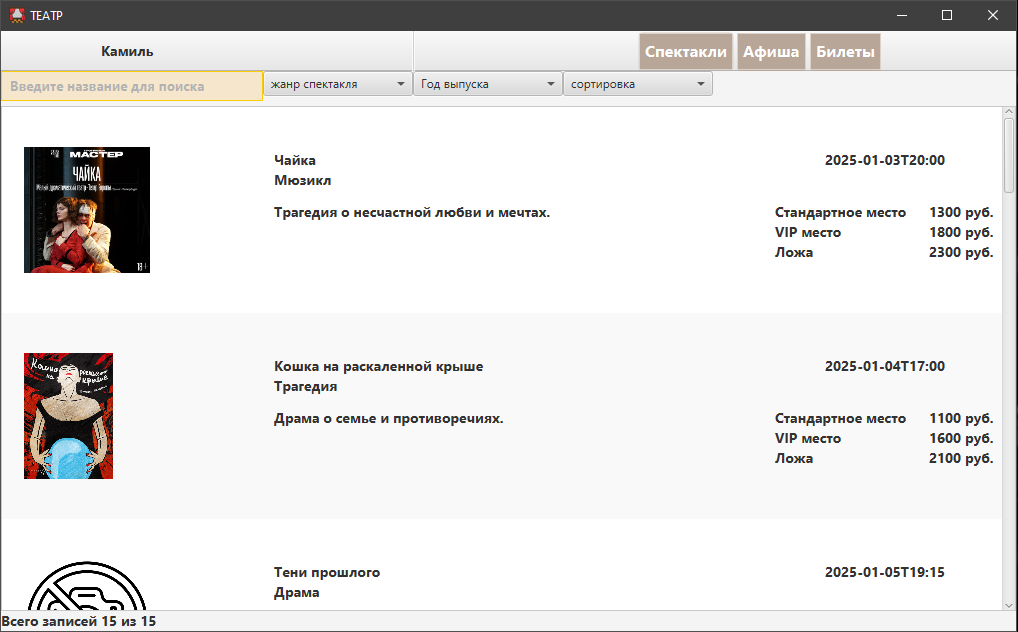


Рис.31 Окно Афиша

Кликнув по кнопке «Выход» программа сначала вернёт пользователя на окно авторизации, если же еще раз нажать на выход, потребует подтверждение. (Рис.32)

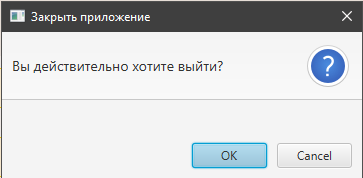


Рис.32 Сообщение программы

Ниже описаны пункты меню управления и информации.

«Страница Спектакли»: эта форма предназначена для отображения списка спектаклей, а также для перехода на дополнительные страницы для редактирования/добавления/удаления информации. (Рис.33)

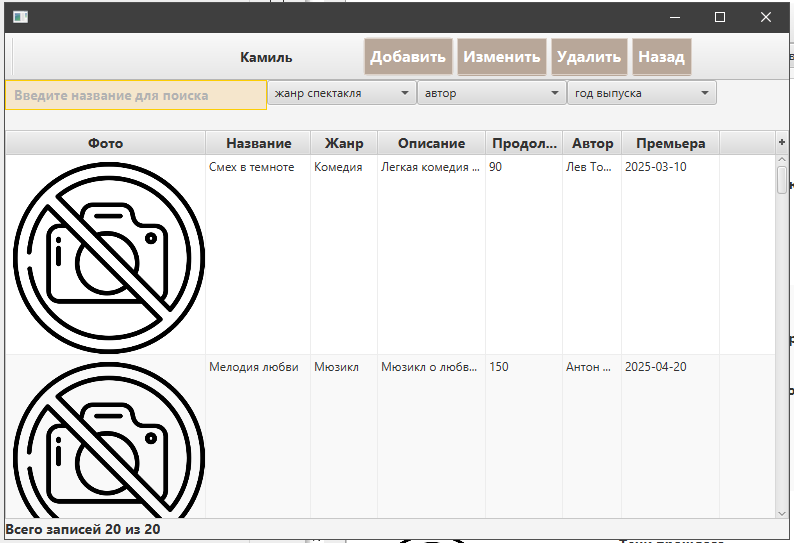


Рис.33 Страница Спектакли

«Страница Добавление и редактирования»: эта форма предназначена для редактирования и добавление данных в таблице Спектакли. (Рис.34)

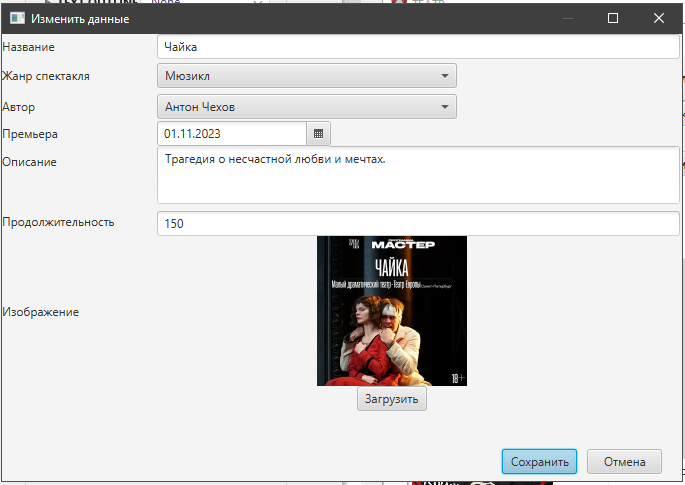


Рис.34 Страница Добавление и редактирование

«Афиша»: эта страница предназначена для отображения списка афиш, а также для перехода на дополнительные страницы для редактирования/добавления/удаления информации. (Рис.35)

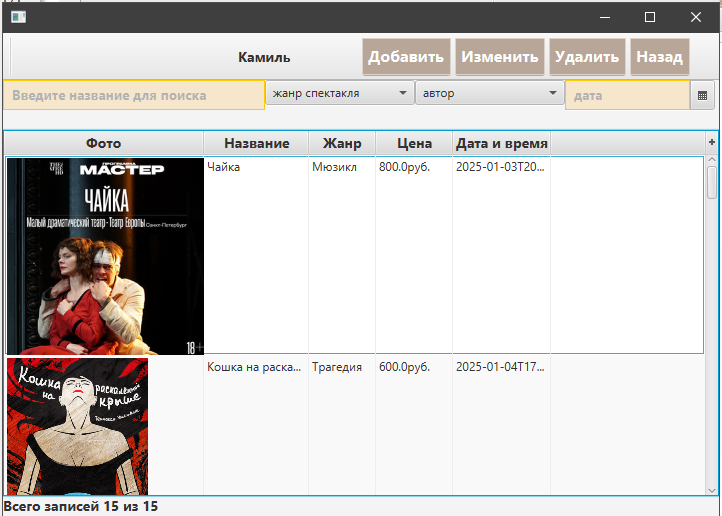


Рис.35 Страница Статусы

«Страница Добавление и редактирования»: эта форма предназначена для редактирования и добавление данных в таблице Афиши. (Рис.36)

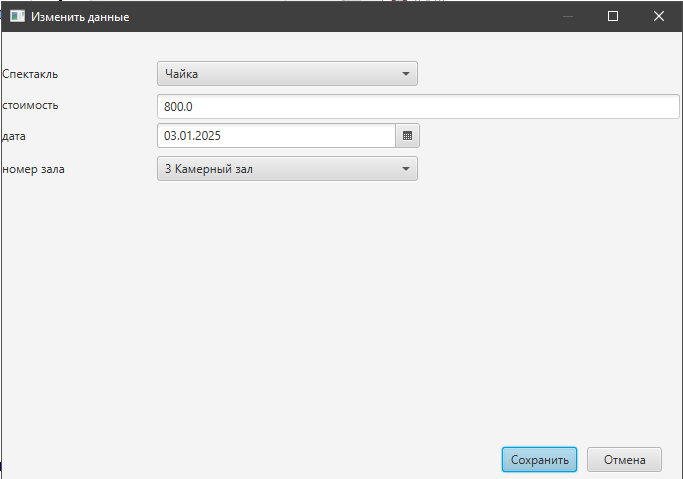


Рис.36 Страница Добавление и редактирование

«Билеты»: эта страница предназначена для вывода информации данных о билетах пользователя. (Рис.37)

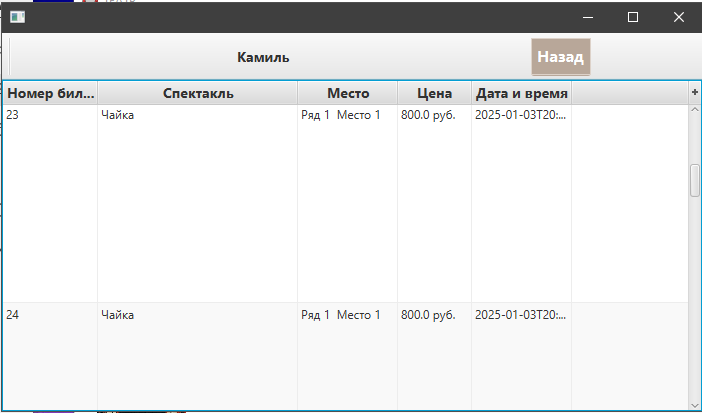


Рис.37 Страница Билеты

«Страница Оформления билета»: эта форма предназначена для добавление данных в таблицу Билеты. (Рис.38)

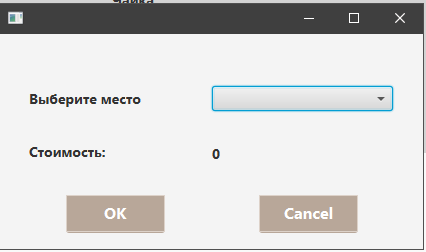


Рис.38 Страница Билеты

Работа с TheaterAp

После запуска программы и прохождения авторизации вы перейдете на главную форму программы. В формах «Спектакли», «Афиши» можно добавить, изменить или удалить все необходимые для работы данные. Нажав соответствующую кнопку и выбрав, при необходимости, элемент, после перехода на одну из перечисленных форм.

Для того, чтобы добавить запись, нажмите кнопку «добавить», введите в редактируемые поля и списки данные и нажмите кнопку сохранить. Для того, чтобы изменить запись, выберите нужный элемент списка, нажмите кнопку «изменить», измените данные в редактируемых полях и нажмите кнопку сохранить.

Для оформления билета необходимо нажать на спектакль, находящийся на главной форме, правой кнопкой мыши, нажать кнопку «добавить билет», затем выбрать место и нажать «сохранить»

После того, как Вы закончите сеанс работы с программой, если Вы хотите выйти из TheaterAp кликните на соответствующей кнопке «Выход».

Основные кнопки TheaterAp

Кнопки управления «Добавить запись», «Удалить запись», «Сохранить» расположены на каждой конкретной форме в базе данных, подразумевающих редактирование.

Действия этих кнопок приведены в таблице 1:

Таблица-1 Действие кнопок

|  |  |
| --- | --- |
| Добавить запись | Прежде чем добавить запись в таблицу необходимо нажать на данную кнопку. Поля формы будут очищены, пользователю требуется ввести необходимые данные. Если пользователь нажмет кнопку «Сохранить данные», то введенная информация добавится в БД. Если нажать кнопку «Отменить», то введенная информация пользователем удалится. |
| Удалить запись/отменить | Кнопка «Удалить запись» позволяет отменить действия при вводе, либо удаляет конкретную запись. |
| Сохранить данные/обновить запись | Кнопка «Сохранить данные» добавляет запись введенную пользователем в БД, либо обновляет существующую. |
| Редактировать | Открывает форму в режиме редактирования информации |
| Поиск | «Поиск» позволяет найти нужную информацию. |
| Фильтрация списков | Позволяют сортировать и фильтровать данные, выводимые из БД, в списках или таблицах |

**4.4 Сообщения пользователю**

Для уверенной и комфортной работы пользователю важно понимать поведение программы, а программе, в свою очередь, поведение пользователя. Для информирования и уточнения намерений пользователей в программах используются сообщения.

При попытке удалить запись, пользователю выводится сообщение с запросом на подтверждение или отмену действия. (Рис.39)

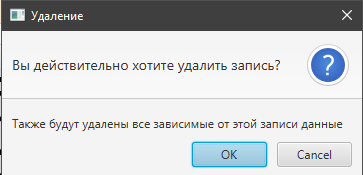


Рис.39 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.40)

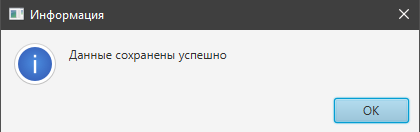


Рис.40 Сообщение Запись добавлена

После обновления записи, пользователю выводится сообщение. (Рис.41)

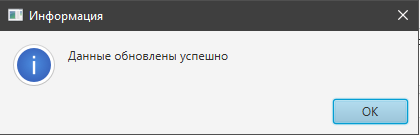


Рис.41 Сообщение Запись обновлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.42)

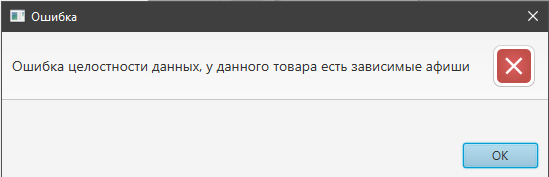


Рис.42 Сообщение Ошибка удаления

**5. РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Обоснование необходимого количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.43 Потоковый граф

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=34, N=25. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=34-25+2=11.

Для данного программного средства необходимо разработать не менее 11 тестовых сценариев.

Чтобы определить количество тестовых прогонов для программного продукта TheaterAp необходимо выделить требования. Из требований к приложению выделим поддержку 2-х операционных систем с одним основным языком локализации и выполнения 6-ти основных функций. Кроме того, приложение позволяет выполнять 5 функций по взаимодействию с системой (запуск приложения, выход из приложения, сохранение результатов в файл, работа с буфером и т.п.).

Таким образом, полное покрытие требований задаёт набор из 2\*1\*(6+5) =22 тестовых прогонов.

**5.2 Описание тестовых пакетов**

Таблица 2 – Тестирование корректного добавления информации в Спектакли

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Спектакли | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Спектакли | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Спектакли открыта | | Отображается форма Спектакли |  |

Таблица 3 – Тестирование некорректного добавления информации в таблицу Спектакли

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Спектакли | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Спектакли | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Но не введем данные в 1 или несколько полей | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Появится всплывающее окно с ошибкой и указанием того, что конкретные поля являются пустыми | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Спектаклиоткрыта | | Отображается форма Спектакли |  |

Таблица 4 – Тестирование корректного редактирования информации в таблице Спектакли

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Спектакли | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Спектакли | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Спектакли  открыта | | Отображается форма Спектакли |  |

Таблица 5 – Тестирование некорректного редактирования информации в таблице Спектакли

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Спектакли | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Спектакли | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите спектакль нажмите кнопку Изменить заполните форму, но не введем данные в 1 или несколько полей | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Появится всплывающее окно с ошибкой и указанием того, что конкретные поля являются пустыми | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Спектакли  открыта | | Отображается форма Спектакли |  |

Таблица 6 – Тестирование корректного добавления информации в таблицу Афиша

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Афиша | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Афиша | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  полностью | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Афиша открыта | | Отображается форма Афиша |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Афиша

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Афиша | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Афиша | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму, но не введем данные в 1 или несколько полей | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Появится всплывающее окно с ошибкой и указанием того, что конкретные поля являются пустыми | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Афиша  открыта | | Отображается форма Афиша |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Афиша

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Афиша | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Афиша | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку изменить Заполните форму полностью | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Афиша  открыта | | Отображается форма Афиша |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Афиша

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Афиша | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Афиша | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| |  | | --- | | Выберите афишу Нажмите кнопку Изменить Заполните форму, но не введем данные в 1 или несколько полей | | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Афиша открыта | | Отображается форма Афиша |  |

Таблица 10 – Тестирование корректного добавления информации в таблице Билеты

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблице Билеты | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Оформление билета | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выбираем место для билета | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Афиши открыта | | Отображается форма Афиши |  |

Таблица 11 – Тестирование некорректного добавления информации в таблицу Билеты

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Билеты | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Билеты | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Оставляем место билета пустым | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Всплывающее окно с ошибкой о пустом поле выбора места | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Афиши открыта | | Отображается форма Афиши |  |

**6. ЗАКЛЮЧЕНИЕ**

**6.1 Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система – Astra Linux.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 2 Gb

Таблица 12 – Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 13 – Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |
| *Тестируемость:*  *-трудоемкость тестирования изменений;*  *- длительность тестирования изменений.* | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

Таблица 14 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 9 | x1 | x2 | **x3** | =27 |
| Отчет | 2 | x2 | **x5** | x8 | =10 |

Таблица 15 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество Категорииских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | **Всего <8** | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| <3 | Простой | Простой | Средний |
| 3-7 | Простой | Средний | Сложный |
| **>8** | Средний | **Сложный** | Сложный |

Таблица 16 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество Категорииских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| 0 или 1 | Простой | Простой | Средний |
| **2 или 3** | Простой | **Средний** | Сложный |
| >4 | Средний | Сложный | Сложный |

Производительность = 4 тысяч строк кода/2 месяца= 2 тысяч строк кода/месяц

Качество = 2 ошибки /4 тысяч строк кода=0,5 ошибок/тысяч строк кода

**7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Г. Шилдт C#: Учебный курс. - Спб.: Питер, 2016. - 512с

2. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2016. - 684 с.

3. Ватсон К. Си Шарп/К.Ватсон.-М.:Лори, 2017.-862 с

4. Балена Ф., Димауро Д. Современная практика программирования на Visual Basic и Visual C#.-М.:Русская редакция,2015,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС,2015,464с.,СД

6. Лабор В.В. Си Шарп: СоЗдание приложений для Windows/-Мн.: Харвест,2016-384с.

7. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2009-512с.

8. ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению

9. ГОСТ 19.102-77.ЕСПД. Стадии разработки

10. ГОСТ 19.505-79.ЕСПД. Руководство оператора. Требования к содержанию и оформлению

11. ГОСТ 19.002-80.ЕСПД. Схемы алгоритмов и программ. Правила выполнения

12. ГОСТ 19.404-79.ЕСПД. Пояснительная записка. Требования к содержанию и оформлению

13. Владимир Биллинг. Основы программирования на С#. 2016 г.

[Электронный ресурс http://mexalib.com/view/42671]

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2017. 2017 г.

[Электронный ресурс http://www.litres.ru/nikita-kultin/osnovy-programmirovaniya-v-microsoft-visual-c-2010-2]

15. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2017.

[Электронный ресурс http://www.twirpx.com/file/27852]

16. Йен Гриффитс. Программирование на С#. 2016 г.

[Электронный ресурс http://rutracker.org/forum/viewtopic.php?t=4772853]

**Приложение А**

Разработка интегрированного программного модуля «Афиша»

Текст кода программного средства

460.ЗМК.14-01 12 01

**Текст кода программного средства**

**Стили приложения**

.root {

-fx-main-bg-color: #FFF;

-fx-additional-bg-color: #ddd2c9;

-fx-accent-bg-color: #FCCD04;

-fx-headers-color: black;

-fx-fields-color: #F5E6CC;

}

.grid-pane {

-fx-background-color: -fx-main-bg-color;

}

.label

{

-fx-font-weight: bold;

-fx-font-size: 14px;

-fx-background-color: transparent;

}

.header-label

{

-fx-font-size: 30px;

-fx-font-weight: bold;

-fx-background-color: transparent;

-fx-text-fill: -fx-headers-color;

}

.button {

-fx-border-style: solid;

-fx-border-color: -fx-additional-bg-color;

-fx-background-color: #b8a799;

-fx-padding: 5 5;

-fx-text-fill: -fx-main-bg-color;

-fx-text-align: center;

-fx-display: inline-block;

-fx-font-size: 16px;

-fx-font-weight: bold;

}

.button:hover {

-fx-background-color: white;

-fx-border-color: none;

-fx-text-fill: #000000;

}

.text-field

{

-fx-font-size: 10pt;

-fx-background-color: -fx-fields-color;

-fx-text-fill: black;

-fx-font-weight: bold;

-fx-border-style: solid; /\* Green \*/

-fx-border-width: 1px; /\* Green \*/

-fx-border-color: -fx-accent-bg-color;

}

**Класс Actor**

package ru.demo.theaterap.model;

import jakarta.persistence.\*;

import javafx.embed.swing.SwingFXUtils;

import javafx.scene.image.Image;

import ru.demo.theaterap.TheatreAp;

import javax.imageio.ImageIO;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.util.Objects;

@Entity

@Table(name = "authors", schema = "theater")

public class Author {

public Author(long id, String firstName, String secondName){

this.authorId = id;

this.firstName = firstName;

this.secondName = secondName;

}

public Author(){

}

@Id

@Column(name = "author\_id", nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long authorId;

@Column(name = "first\_name", nullable = false)

private String firstName;

@Column(name = "family", nullable = false)

private String secondName;

@Column(name = "biografy", nullable = false)

private String biografy;

@Column(name = "photo", nullable = false)

private byte[] photo;

public Long getAuthorId() {

return authorId;

}

public void setAuthorId(Long authorId) {

this.authorId = authorId;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getSecondName() {

return secondName;

}

public void setSecondName(String secondName) {

this.secondName = secondName;

}

public String getBiografy() {

return biografy;

}

public void setBiografy(String biografy) {

this.biografy = biografy;

}

public Image getPhoto() throws IOException {

if (photo == null)

return new Image(Objects.requireNonNull(TheatreAp.class.getResourceAsStream("ru/demo/theaterap/NaN.png")));

BufferedImage capture = ImageIO.read(new ByteArrayInputStream(photo));

return SwingFXUtils.toFXImage(capture, null);

}

public void setPhoto(Image img) throws IOException {

BufferedImage buf = SwingFXUtils.fromFXImage(img, null);

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(buf, "jpg", baos);

byte[] bytes = baos.toByteArray();

this.photo = bytes;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Author author = (Author) o;

return Objects.equals(authorId, author.authorId) && Objects.equals(firstName, author.firstName) && Objects.equals(secondName, author.secondName) && Objects.equals(biografy, author.biografy) && Objects.equals(photo, author.photo);

}

@Override

public int hashCode() {

return Objects.hash(authorId, firstName, secondName, biografy);

}

@Override

public String toString() {

return firstName + ' ' + secondName

;

}

**}**

**Класс Жанр**

package ru.demo.theaterap.model;

import jakarta.persistence.\*;

import java.util.Objects;

@Entity

@Table(name = "genres", schema = "theater")

public class Genre {

@Id

@Column(name = "genre\_id", nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long GenreId;

@Column(name = "title", nullable = false)

private String title;

public Genre(long l, String title) {

this.GenreId = l;

this.title = title;

}

public Long getGenreId() {

return GenreId;

}

public void setGenreId(Long genreId) {

GenreId = genreId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Genre genre = (Genre) o;

return Objects.equals(GenreId, genre.GenreId) && Objects.equals(title, genre.title);

}

@Override

public int hashCode() {

return Objects.hash(GenreId, title);

}

@Override

public String toString() {

return title;

}

}

**Класс Зал**

package ru.demo.theaterap.model;

import jakarta.persistence.\*;

import java.util.Objects;

@Entity

@Table(name = "halls", schema = "theater")

public class Hall {

@Id

@Column(name = "hall\_id", nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long HallId;

@Column(name = "number\_hall", nullable = false)

private int NumberHall;

@Column(name = "count\_places", nullable = false)

private int CountPlaces;

@ManyToOne

@JoinColumn(name = "hall\_type", nullable = false)

private HallType HallType;

public Long getHallId() {

return HallId;

}

public void setHallId(Long hallId) {

HallId = hallId;

}

public int getNumberHall() {

return NumberHall;

}

public void setNumberHall(int numberHall) {

NumberHall = numberHall;

}

public int getCountPlaces() {

return CountPlaces;

}

public void setCountPlaces(int countPlaces) {

CountPlaces = countPlaces;

}

public ru.demo.theaterap.model.HallType getHallType() {

return HallType;

}

public void setHallType(ru.demo.theaterap.model.HallType hallType) {

HallType = hallType;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Hall hall = (Hall) o;

return Objects.equals(HallId, hall.HallId) && Objects.equals(NumberHall, hall.NumberHall) && Objects.equals(CountPlaces, hall.CountPlaces) && Objects.equals(HallType, hall.HallType);

}

@Override

public int hashCode() {

return Objects.hash(HallId, NumberHall);

}

@Override

public String toString() {

return NumberHall + " " + HallType;

}

}

**Класс Спектакль**

package ru.demo.theaterap.model;

import jakarta.persistence.\*;

import javafx.embed.swing.SwingFXUtils;

import javafx.scene.image.Image;

import ru.demo.theaterap.TheatreAp;

import javax.imageio.ImageIO;

import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.time.LocalDate;

import java.util.Objects;

@Entity

@Table(name = "performances", schema = "theater")

public class Performance {

@Id

@Column(name = "performance\_id", nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long performanceId;

@Column(name = "title", nullable = false)

private String title;

@ManyToOne

@JoinColumn(name = "genre", nullable = false)

private Genre genre;

@Column(name = "description", nullable = false)

private String description;

@Column(name = "longs", nullable = false)

private int longs;

@ManyToOne

@JoinColumn(name = "author\_id", nullable = false)

private Author authorId;

@Column(name = "premier", nullable = false)

private LocalDate premier;

@Column(name = "photo")

private byte[] photo;

public Long getPerformanceId() {

return performanceId;

}

public void setPerformanceId(Long performanceId) {

this.performanceId = performanceId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Genre getGenre() {

return genre;

}

public void setGenre(Genre genre) {

this.genre = genre;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public int getLongs() {

return longs;

}

public void setLongs(int longs) {

this.longs = longs;

}

public Author getAuthor() {

return authorId;

}

public void setAuthor(Author authorId) {

this.authorId = authorId;

}

public LocalDate getPremier() {

return premier;

}

public void setPremier(LocalDate premier) {

this.premier = premier;

}

public Image getPhoto() throws IOException {

if (photo == null) {

return new Image(Objects.requireNonNull(TheatreAp.class.getResourceAsStream("NaN.png")));

}

else {

BufferedImage capture = ImageIO.read(new ByteArrayInputStream(photo));

return SwingFXUtils.toFXImage(capture, null);

}

}

public void setPhoto(Image img) throws IOException {

BufferedImage buf = SwingFXUtils.fromFXImage(img, null);

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(buf, "jpg", baos);

byte[] bytes = baos.toByteArray();

this.photo = bytes;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Performance that = (Performance) o;

return Objects.equals(performanceId, that.performanceId) && Objects.equals(title, that.title)

&& Objects.equals(genre, that.genre) && Objects.equals(description, that.description) &&

Objects.equals(longs, that.longs) && Objects.equals(authorId, that.authorId) &&

Objects.equals(premier, that.premier) && Objects.deepEquals(photo, that.photo);

}

@Override

public int hashCode() {

return Objects.hash(performanceId, title, premier);

}

@Override

public String toString() {

return title;

}

}

**Класс Расписание**

package ru.demo.theaterap.model;

import jakarta.persistence.\*;

import java.time.LocalDateTime;

import java.util.Objects;

@Entity

@Table(name = "schedules", schema = "theater")

public class Schedule {

@Id

@Column(name = "schedule\_id", nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long scheduleId;

@ManyToOne

@JoinColumn(name = "performance\_id", nullable = false)

private Performance performanceId;

@ManyToOne

@JoinColumn(name = "hall\_id", nullable = false)

private Hall hallId;

@Column(name = "date\_start", nullable = false)

private LocalDateTime dateStart;

@Column(name = "price", nullable = false)

private double price;

public Long getScheduleId() {

return scheduleId;

}

public void setScheduleId(Long scheduleId) {

this.scheduleId = scheduleId;

}

public Performance getPerformanceId() {

return performanceId;

}

public void setPerformanceId(Performance performanceId) {

this.performanceId = performanceId;

}

public Hall getHall() {

return hallId;

}

public void setHall(Hall hallId) {

this.hallId = hallId;

}

public LocalDateTime getDateStart() {

return dateStart;

}

public void setDateStart(LocalDateTime dateStart) {

this.dateStart = dateStart;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Schedule schedules = (Schedule) o;

return Double.compare(price, schedules.price) == 0 && Objects.equals(scheduleId, schedules.scheduleId) && Objects.equals(performanceId, schedules.performanceId) && Objects.equals(hallId, schedules.hallId) && Objects.equals(dateStart, schedules.dateStart);

}

@Override

public int hashCode() {

return Objects.hash(scheduleId, dateStart, price);

}

@Override

public String toString() {

return "Schedules{"+

"dateStart='" + dateStart + '\'' +

", price=" + price +

'}';

}

}

**Форма Главная**

**Код интерфейса**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.PasswordField?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.ColumnConstraints?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.RowConstraints?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="300.0" maxWidth="500.0" minHeight="-Infinity" minWidth="-Infinity" prefHeight="300.0"

prefWidth="500.0" style="-fx-background-color: #eee8e4;" xmlns="http://javafx.com/javafx/23.0.1"

xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.demo.theaterap.controller.LoginController">

<children>

<GridPane alignment="CENTER" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="293.0" minWidth="10.0" prefWidth="251.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="250.0" minWidth="10.0" prefWidth="249.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="76.0" minHeight="10.0" prefHeight="76.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="105.0" minHeight="10.0" prefHeight="64.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="105.0" minHeight="10.0" prefHeight="74.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="99.0" minHeight="10.0" prefHeight="87.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<PasswordField fx:id="PasswordField" text="1" prefHeight="40.0" prefWidth="462.0"

promptText="Пароль" GridPane.columnSpan="2" GridPane.rowIndex="2">

<GridPane.margin>

<Insets left="20.0" right="20.0" />

</GridPane.margin>

</PasswordField>

<Button fx:id="BtnOK" onAction="#btnOkOnAction" alignment="CENTER" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" mnemonicParsing="false" prefHeight="40.0" prefWidth="160.0" text="ОK" GridPane.rowIndex="3">

<GridPane.margin>

<Insets bottom="20.0" left="30.0" right="30.0" top="20.0" />

</GridPane.margin>

</Button>

<TextField fx:id="TextFieldUserName" text="ret" prefHeight="40.0" prefWidth="388.0" promptText="Логин" GridPane.columnSpan="2" GridPane.rowIndex="1">

<GridPane.margin>

<Insets left="20.0" right="20.0" />

</GridPane.margin>

</TextField>

<Button fx:id="BtnCancel" onAction="#btnCancelOnAction" alignment="CENTER" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" mnemonicParsing="false" prefHeight="40.0" prefWidth="157.0" text="Cancel" GridPane.columnIndex="1" GridPane.rowIndex="3">

<GridPane.margin>

<Insets bottom="20.0" left="30.0" right="30.0" top="20.0" />

</GridPane.margin>

</Button>

<Label alignment="CENTER" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" prefHeight="75.0" prefWidth="205.0" style="-fx-background-color: #e6ddd6;" styleClass="header-label" text="Theater" textAlignment="CENTER" GridPane.columnSpan="2">

<font>

<Font size="30.0" />

</font>

</Label>

<ImageView fitHeight="50.0" fitWidth="60.0" pickOnBounds="true" preserveRatio="true">

<GridPane.margin>

<Insets left="30.0" />

</GridPane.margin>

<image>

<!-- <Image url="/ru/demo/theaterap/theaterIcon.png" />-->

</image>

</ImageView>

</children>

</GridPane>

</children>

</AnchorPane>

**Код Контроллера для страницы**

package ru.demo.theaterap.controller;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.stage.Stage;

import ru.demo.theaterap.TheatreAp;

import ru.demo.theaterap.model.Genre;

import ru.demo.theaterap.model.Schedule;

import ru.demo.theaterap.service.GenreService;

import ru.demo.theaterap.service.ScheduleService;

import ru.demo.theaterap.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.util.\*;

import java.util.stream.Collectors;

public class MainWindowController implements Initializable {

public ListView ListViewPerformance;

public TextField TextFieldSearch;

public ComboBox ComboBoxGenre;

public ComboBox ComboboxSort;

public Button BtnPerformance;

public Button BtnBack;

public Label LabelInfo;

public ComboBox ComboBoxYear;

public Button BtnSchedule;

private ScheduleService scheduleService = new ScheduleService();

private int itemsCount;

@FXML

private Label LabelUser;

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

LabelUser.setText(Manager.currentUser.getFirstName());

List<Genre> genreList = GenreService.findAll();

genreList.add(0, new Genre(0L, "Все"));

ObservableList<Genre> genres = FXCollections.observableArrayList(genreList);

ComboBoxGenre.setItems(genres);

ObservableList<String> performances = FXCollections.observableArrayList("все", "по возрастанию цены", "по убыванию цены");

ComboboxSort.setItems(performances);

Set<Integer> years = ScheduleService.findAll().stream().map(schedule -> schedule.getPerformanceId().getPremier().getYear()).collect(Collectors.toSet());

List<Integer> sortedYears = new ArrayList<>(years);

Collections.sort(sortedYears, Collections.reverseOrder());

List<String> yearOptions = new ArrayList<>();

yearOptions.add("Все годa");

sortedYears.forEach(year -> yearOptions.add(String.valueOf(year)));

ObservableList<String> yearsList = FXCollections.observableArrayList(yearOptions);

ComboBoxYear.setItems(yearsList);

filterData();

}

public void TextFieldSearchAction(ActionEvent actionEvent) {

filterData();

}

public void ComboBoxGenreAction(ActionEvent actionEvent) {

filterData();

}

public void ComboboxSortAction(ActionEvent actionEvent) {

filterData();

}

public void BtnPerformanceAction(ActionEvent actionEvent) {

FXMLLoader fxmlLoader = new FXMLLoader(TheatreAp.class.getResource("performance-table-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("style.css");

Manager.secondStage = new Stage();

Manager.secondStage.setScene(scene);

Manager.secondStage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

public void BtnTicketAction(ActionEvent actionEvent) {

FXMLLoader fxmlLoader = new FXMLLoader(TheatreAp.class.getResource("ticket-table-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("style.css");

Manager.secondStage = new Stage();

Manager.secondStage.setScene(scene);

Manager.secondStage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

void filterData() {

List<Schedule> schedules = scheduleService.findAll();

itemsCount = schedules.size();

if (!ComboBoxGenre.getSelectionModel().isEmpty()) {

Genre genre = (Genre) ComboBoxGenre.getValue();

if (genre != null && genre.getGenreId() != null && genre.getGenreId() != 0L) {

schedules = schedules.stream().filter(schedule -> schedule.getPerformanceId().getGenre().getGenreId().equals(genre.getGenreId())).collect(Collectors.toList());

}

}

if (!ComboboxSort.getSelectionModel().isEmpty()) {

String order = (String) ComboboxSort.getValue();

if (order.equals("по возрастанию цены")) {

schedules = schedules.stream().sorted(Comparator.comparing(Schedule::getPrice)).collect(Collectors.toList());

}

if (order.equals("по убыванию цены")) {

schedules = schedules.stream().sorted(Comparator.comparing(Schedule::getPrice)).collect(Collectors.toList()).reversed();

}

}

if (!ComboBoxYear.getSelectionModel().isEmpty()) {

String selectedYear = (String) ComboBoxYear.getValue();

if (selectedYear != null && !selectedYear.equals("Все годa")) {

int year = Integer.parseInt(selectedYear);

schedules = schedules.stream().filter(schedule -> schedule.getPerformanceId().getPremier().getYear() == year).collect(Collectors.toList());

}

}

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

schedules = schedules.stream().filter(schedule -> schedule.getPerformanceId().getTitle().toLowerCase().contains(searchText.toLowerCase())).collect(Collectors.toList());

}

ListViewPerformance.getItems().clear();

for (Schedule schedule : schedules) {

ListViewPerformance.getItems().add(schedule);

}

ListViewPerformance.setCellFactory(lv -> new ScheduleCell());

int filteredItemsCount = schedules.size();

LabelInfo.setText("Всего записей " + filteredItemsCount + " из " + itemsCount);

}

public void ComboBoxYearAction(ActionEvent actionEvent) {

filterData();

}

public void BtnScheduleAction(ActionEvent actionEvent) {

FXMLLoader fxmlLoader = new FXMLLoader(TheatreAp.class.getResource("schedule-table-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("style.css");

Manager.secondStage = new Stage();

Manager.secondStage.setScene(scene);

Manager.secondStage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

}

**Страница Спектакль**

**Код интерфейса**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Separator?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.control.ToolBar?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.FlowPane?>

<AnchorPane prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.demo.theaterap.controller.PerformanceTableView">

<children>

<BorderPane prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

<top>

<ToolBar prefHeight="40.0" prefWidth="200.0" BorderPane.alignment="CENTER\_RIGHT">

<items>

<Separator halignment="LEFT" maxWidth="1.7976931348623157E308" prefHeight="30.0" prefWidth="225.0" />

<Label fx:id="LabelUser" prefWidth="120.0" text="Label" />

<Button fx:id="BtnAdd" mnemonicParsing="false" onAction="#BtnAddAction" text="Добавить" />

<Button fx:id="BtnUpdate" mnemonicParsing="false" onAction="#BtnUpdateAction" text="Изменить" />

<Button fx:id="BtnDelete" mnemonicParsing="false" onAction="#BtnDeleteAction" text="Удалить" />

<Button fx:id="BtnBack" alignment="CENTER\_RIGHT" mnemonicParsing="false" onAction="#BtnBackAction" text="Назад" textAlignment="RIGHT" />

</items>

</ToolBar>

</top>

<center>

<BorderPane BorderPane.alignment="CENTER">

<top>

<FlowPane minHeight="-Infinity" nodeOrientation="LEFT\_TO\_RIGHT" prefHeight="50.0" prefWidth="200.0" rowValignment="TOP" BorderPane.alignment="CENTER">

<children>

<TextField fx:id="TextFieldSearch" onAction="#TextFieldSearchAction" onInputMethodTextChanged="#TextFieldTextChanged" prefHeight="25.0" prefWidth="262.0" promptText="Введите название для поиска" />

<ComboBox fx:id="ComboBoxGenre" onAction="#ComboBoxGenreAction" prefWidth="150.0" promptText="жанр спектакля" />

<ComboBox fx:id="ComboBoxAuthor" onAction="#ComboBoxAuthorAction" prefWidth="150.0" promptText="автор" />

<ComboBox fx:id="ComboBoxYear" onAction="#ComboBoxYearAction" prefWidth="150.0" promptText="год выпуска" />

</children>

</FlowPane>

</top>

<center>

<TableView fx:id="TableViewPerformance" fixedCellSize="200.0" prefHeight="200.0" prefWidth="200.0" tableMenuButtonVisible="true" BorderPane.alignment="CENTER">

<columns>

<TableColumn id="TableColumnPhoto" fx:id="TableColumnPhoto" minWidth="200.0" prefWidth="200.0" resizable="false" text="Фото" />

<TableColumn id="TableColumnTitle" fx:id="TableColumnTitle" maxWidth="1.7976931348623157E308" prefWidth="105.0" text="Название" />

<TableColumn id="TableColumnGenre" fx:id="TableColumnGenre" minWidth="0.0" prefWidth="67.0" text="Жанр" />

<TableColumn id="TableColumnDescription" fx:id="TableColumnDescription" prefWidth="108.0" text="Описание" />

<TableColumn fx:id="TableColumnLongs" prefWidth="77.0" text="Продолжительность" />

<TableColumn fx:id="TableColumnAuthor" prefWidth="59.0" text="Автор" />

<TableColumn fx:id="TableColumnPremier" maxWidth="1.7976931348623157E308" prefWidth="98.0" text="Премьера" />

</columns>

</TableView>

</center>

</BorderPane>

</center>

<bottom>

<Label fx:id="LabelInfo" text="Label" BorderPane.alignment="CENTER\_LEFT" />

</bottom>

</BorderPane>

</children>

</AnchorPane>

**Код Контроллера для страницы**

package ru.demo.theaterap.controller;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleObjectProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.image.ImageView;

import javafx.scene.input.InputMethodEvent;

import javafx.stage.Modality;

import javafx.stage.Stage;

import ru.demo.theaterap.TheatreAp;

import ru.demo.theaterap.model.Author;

import ru.demo.theaterap.model.Genre;

import ru.demo.theaterap.model.Performance;

import ru.demo.theaterap.model.Schedule;

import ru.demo.theaterap.service.AuthorService;

import ru.demo.theaterap.service.GenreService;

import ru.demo.theaterap.service.PerformanceService;

import ru.demo.theaterap.service.ScheduleService;

import ru.demo.theaterap.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.util.\*;

import java.util.stream.Collectors;

import static ru.demo.theaterap.util.Manager.ShowConfirmPopup;

public class PerformanceTableView implements Initializable {

@FXML

private PerformanceService performanceService = new PerformanceService();

@FXML

private TableView TableViewPerformance;

@FXML

private TextField TextFieldSearch;

@FXML

private ComboBox ComboBoxGenre;

@FXML

private ComboBox ComboBoxAuthor;

@FXML

private TableColumn<Performance, ImageView> TableColumnPhoto;

@FXML

private TableColumn<Performance, String> TableColumnTitle;

@FXML

private TableColumn<Performance, String> TableColumnGenre;

@FXML

private TableColumn<Performance, String> TableColumnDescription;

@FXML

private TableColumn<Performance, Integer> TableColumnLongs;

@FXML

private TableColumn<Performance, String> TableColumnAuthor;

@FXML

private TableColumn<Performance, String> TableColumnPremier;

@FXML

private Label LabelInfo;

@FXML

private Label LabelUser;

@FXML

private ScheduleService scheduleService;

@FXML

private AuthorService authorService;

@FXML

private ComboBox ComboBoxYear;

@FXML

private Button BtnBack;

private int itemsCount;

public void BtnAddAction(ActionEvent actionEvent) {

Performance performance = (Performance) TableViewPerformance.getSelectionModel().getSelectedItem();

Manager.currentPerformance = null;

ShowEditProductWindow();

filterData();

}

public void BtnUpdateAction(ActionEvent actionEvent) {

Performance performance = (Performance) TableViewPerformance.getSelectionModel().getSelectedItem();

if (performance != null){

Manager.currentPerformance = performance;

ShowEditProductWindow();

filterData();

}

else

Manager.showErrorMessageBox("Не выбран спектакль для редактирования");

}

public void BtnDeleteAction(ActionEvent actionEvent) {

Performance performance = (Performance) TableViewPerformance.getSelectionModel().getSelectedItem();

List<Schedule> schedules = scheduleService.findAll();

if (performance != null)

{

if (schedules.stream().anyMatch(schedule -> schedule.getPerformanceId().getPerformanceId().equals(performance.getPerformanceId()))){

Manager.showErrorMessageBox("Ошибка целостности данных, у данного товара есть зависимые афиши");

return;

}

Optional<ButtonType> result = ShowConfirmPopup();

if (result.get() == ButtonType.OK) {

performanceService.delete(performance);

filterData();

}

}

else{

Manager.showErrorMessageBox("Не выбран спектакль для удаления");

return;

}

}

void ShowEditProductWindow() {

Stage newWindow = new Stage();

FXMLLoader fxmlLoader = new FXMLLoader(TheatreAp.class.getResource("performance-edit-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

} catch (IOException e) {

throw new RuntimeException(e);

}

newWindow.setTitle("Изменить данные");

newWindow.initOwner(Manager.secondStage);

newWindow.initModality(Modality.WINDOW\_MODAL);

newWindow.setScene(scene);

Manager.currentStage = newWindow;

newWindow.showAndWait();

Manager.currentStage = null;

filterData();

}

public void BtnBackAction(ActionEvent actionEvent) {

Stage stage = (Stage) BtnBack.getScene().getWindow();

stage.close();

}

public void TextFieldTextChanged(InputMethodEvent inputMethodEvent) {

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

initController();

}

public void initController() {

LabelUser.setText(Manager.currentUser.getFirstName());

List<Genre> genreList = GenreService.findAll();

genreList.add(0, new Genre(0L, "Все жанры"));

ObservableList<Genre> genres = FXCollections.observableArrayList(genreList);

ComboBoxGenre.setItems(genres);

List<Author> authorList = AuthorService.findAll();

authorList.add(0, new Author(0L, "Все ", "авторы"));

ObservableList<Author> authors = FXCollections.observableArrayList(authorList);

ComboBoxAuthor.setItems(authors);

Set<Integer> years = ScheduleService.findAll().stream().map(schedule -> schedule.getPerformanceId().getPremier().getYear()).collect(Collectors.toSet());

List<Integer> sortedYears = new ArrayList<>(years);

Collections.sort(sortedYears, Collections.reverseOrder());

List<String> yearOptions = new ArrayList<>();

yearOptions.add("Все годa");

sortedYears.forEach(year -> yearOptions.add(String.valueOf(year)));

ObservableList<String> yearsList = FXCollections.observableArrayList(yearOptions);

ComboBoxYear.setItems(yearsList);

setCellValueFactories();

filterData();

}

private void setCellValueFactories() {

TableColumnPhoto.setCellValueFactory(cellData -> {

try {

ImageView imageView = new ImageView(cellData.getValue().getPhoto());

imageView.setFitWidth(200);

imageView.setFitHeight(200);

imageView.setPreserveRatio(true);

return new SimpleObjectProperty<ImageView>(imageView);

} catch (IOException e) {

throw new RuntimeException(e);

}

});

TableColumnTitle.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getTitle()));

TableColumnGenre.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getGenre().getTitle()));

TableColumnDescription.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getDescription()));

TableColumnLongs.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getLongs()).asObject());

TableColumnAuthor.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getAuthor().getFirstName() + ' ' + cellData.getValue().getAuthor().getSecondName()));

TableColumnPremier.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getPremier().toString()));

}

@FXML

void TextFieldTextChanged(ActionEvent event) {

filterData();

}

public void TextFieldSearchAction(ActionEvent actionEvent) {

filterData();

}

public void ComboBoxGenreAction(ActionEvent actionEvent) {filterData();}

public void ComboBoxAuthorAction(ActionEvent actionEvent) {filterData();}

public void ComboBoxYearAction(ActionEvent actionEvent) {filterData();}

void filterData() {

List<Performance> performances = performanceService.findAll();

itemsCount = performances.size();

if (!ComboBoxGenre.getSelectionModel().isEmpty()) {

Genre genre = (Genre) ComboBoxGenre.getValue();

if (genre != null && genre.getGenreId() != null && genre.getGenreId() != 0L) {

performances = performances.stream().filter(performance -> performance.getGenre().getGenreId().equals(genre.getGenreId())).collect(Collectors.toList());

}

}

if (!ComboBoxAuthor.getSelectionModel().isEmpty()){

Author author = (Author) ComboBoxAuthor.getValue();

if (author.getAuthorId() != null && author.getAuthorId() != 0L) {

performances = performances.stream().filter(performance -> performance.getAuthor().getAuthorId().equals(author.getAuthorId())).collect(Collectors.toList());

}

}

if (!ComboBoxYear.getSelectionModel().isEmpty()) {

String selectedYear = (String) ComboBoxYear.getValue();

if (selectedYear != null && !selectedYear.equals("Все годa")) {

int year = Integer.parseInt(selectedYear);

performances = performances.stream().filter(performance -> performance.getPremier().getYear() == year).collect(Collectors.toList());

}

}

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

performances = performances.stream().filter(performance -> performance.getTitle().toLowerCase().contains(searchText.toLowerCase())).collect(Collectors.toList());

}

TableViewPerformance.getItems().clear();

for (Performance performance : performances) {

TableViewPerformance.getItems().add(performance);

}

int filteredItemsCount = performances.size();

LabelInfo.setText("Всего записей " + filteredItemsCount + " из " + itemsCount);

}

}

**Страница Редактирования спектакля**

**Файл интерфейса**

<Page x:Class="AfishaApp.Pages.EventsPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:AfishaApp.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Мероприятия" IsVisibleChanged="PageIsVisibleChanged" Style="{StaticResource base\_page}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="40"/>

</Grid.RowDefinitions>

<DataGrid x:Name="DataGridGood" Grid.Row="0"

AutoGenerateColumns="False"

IsReadOnly="True"

RowHeight="30"

SelectionMode="Single"

ItemContainerStyle="{StaticResource grid\_item}"

LoadingRow="DataGridGoodLoadingRow">

<DataGrid.Columns>

<DataGridTemplateColumn Width="100" Header="Изображение">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Image Source="{Binding GetPhoto}" />

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Binding="{Binding EventName}" Header="Название" Width="2\*"/>

<DataGridTextColumn Binding="{Binding Info}" Header="Информация" Width="\*"/>

<DataGridTextColumn Binding="{Binding EventDate, StringFormat='d'}" Header="Дата проведения" Width="\*"/>

<DataGridTextColumn Binding="{Binding EventPlace}" Header="Место проведения" Width="\*"/>

<DataGridTextColumn Binding="{Binding Duration}" Header="Длительность" Width="\*"/>

<DataGridTextColumn Binding="{Binding Category.CategoryName}" Header="Категория" Width="\*"/>

<DataGridTextColumn Binding="{Binding Status.StatusName}" Header="Статус" Width="\*"/>

<DataGridTemplateColumn Width="auto" Header="Редактировать">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnEdit" Content="..." Click="ButtonClick" Margin="0"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Button x:Name="BtnAdd" Grid.Row="1" Width="100" HorizontalAlignment="Right" Margin="3 0 120 0"

Content="Добавить" Click="BtnAddClick"/>

<Button x:Name="BtnDelete" Grid.Row="1" Width="100" HorizontalAlignment="Right" Margin="3"

Content="Удалить" Click="BtnDeleteClick"/>

<Button x:Name="BtnEditStatus" Grid.Row="1" Width="100" HorizontalAlignment="Left" Content="Статусы" Click="BtnEditStatusClick"/>

<Button x:Name="BtnEditCategory" Grid.Row="1" Width="100" HorizontalAlignment="Left" Margin="120 3 0 0" Content="Категории" Click="BtnEditCategoryClick"/>

</Grid>

</Page>

**Код Контроллера для страницы**

package ru.demo.theaterap.controller;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleObjectProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.image.ImageView;

import javafx.scene.input.InputMethodEvent;

import javafx.stage.Modality;

import javafx.stage.Stage;

import ru.demo.theaterap.TheatreAp;

import ru.demo.theaterap.model.Author;

import ru.demo.theaterap.model.Genre;

import ru.demo.theaterap.model.Performance;

import ru.demo.theaterap.model.Schedule;

import ru.demo.theaterap.service.AuthorService;

import ru.demo.theaterap.service.GenreService;

import ru.demo.theaterap.service.PerformanceService;

import ru.demo.theaterap.service.ScheduleService;

import ru.demo.theaterap.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.util.\*;

import java.util.stream.Collectors;

import static ru.demo.theaterap.util.Manager.ShowConfirmPopup;

public class PerformanceTableView implements Initializable {

@FXML

private PerformanceService performanceService = new PerformanceService();

@FXML

private TableView TableViewPerformance;

@FXML

private TextField TextFieldSearch;

@FXML

private ComboBox ComboBoxGenre;

@FXML

private ComboBox ComboBoxAuthor;

@FXML

private TableColumn<Performance, ImageView> TableColumnPhoto;

@FXML

private TableColumn<Performance, String> TableColumnTitle;

@FXML

private TableColumn<Performance, String> TableColumnGenre;

@FXML

private TableColumn<Performance, String> TableColumnDescription;

@FXML

private TableColumn<Performance, Integer> TableColumnLongs;

@FXML

private TableColumn<Performance, String> TableColumnAuthor;

@FXML

private TableColumn<Performance, String> TableColumnPremier;

@FXML

private Label LabelInfo;

@FXML

private Label LabelUser;

@FXML

private ScheduleService scheduleService;

@FXML

private AuthorService authorService;

@FXML

private ComboBox ComboBoxYear;

@FXML

private Button BtnBack;

private int itemsCount;

public void BtnAddAction(ActionEvent actionEvent) {

Performance performance = (Performance) TableViewPerformance.getSelectionModel().getSelectedItem();

Manager.currentPerformance = null;

ShowEditProductWindow();

filterData();

}

public void BtnUpdateAction(ActionEvent actionEvent) {

Performance performance = (Performance) TableViewPerformance.getSelectionModel().getSelectedItem();

if (performance != null){

Manager.currentPerformance = performance;

ShowEditProductWindow();

filterData();

}

else

Manager.showErrorMessageBox("Не выбран спектакль для редактирования");

}

public void BtnDeleteAction(ActionEvent actionEvent) {

Performance performance = (Performance) TableViewPerformance.getSelectionModel().getSelectedItem();

List<Schedule> schedules = scheduleService.findAll();

if (performance != null)

{

if (schedules.stream().anyMatch(schedule -> schedule.getPerformanceId().getPerformanceId().equals(performance.getPerformanceId()))){

Manager.showErrorMessageBox("Ошибка целостности данных, у данного товара есть зависимые афиши");

return;

}

Optional<ButtonType> result = ShowConfirmPopup();

if (result.get() == ButtonType.OK) {

performanceService.delete(performance);

filterData();

}

}

else{

Manager.showErrorMessageBox("Не выбран спектакль для удаления");

return;

}

}

void ShowEditProductWindow() {

Stage newWindow = new Stage();

FXMLLoader fxmlLoader = new FXMLLoader(TheatreAp.class.getResource("performance-edit-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

} catch (IOException e) {

throw new RuntimeException(e);

}

newWindow.setTitle("Изменить данные");

newWindow.initOwner(Manager.secondStage);

newWindow.initModality(Modality.WINDOW\_MODAL);

newWindow.setScene(scene);

Manager.currentStage = newWindow;

newWindow.showAndWait();

Manager.currentStage = null;

filterData();

}

public void BtnBackAction(ActionEvent actionEvent) {

Stage stage = (Stage) BtnBack.getScene().getWindow();

stage.close();

}

public void TextFieldTextChanged(InputMethodEvent inputMethodEvent) {

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

initController();

}

public void initController() {

LabelUser.setText(Manager.currentUser.getFirstName());

List<Genre> genreList = GenreService.findAll();

genreList.add(0, new Genre(0L, "Все жанры"));

ObservableList<Genre> genres = FXCollections.observableArrayList(genreList);

ComboBoxGenre.setItems(genres);

List<Author> authorList = AuthorService.findAll();

authorList.add(0, new Author(0L, "Все ", "авторы"));

ObservableList<Author> authors = FXCollections.observableArrayList(authorList);

ComboBoxAuthor.setItems(authors);

Set<Integer> years = ScheduleService.findAll().stream().map(schedule -> schedule.getPerformanceId().getPremier().getYear()).collect(Collectors.toSet());

List<Integer> sortedYears = new ArrayList<>(years);

Collections.sort(sortedYears, Collections.reverseOrder());

List<String> yearOptions = new ArrayList<>();

yearOptions.add("Все годa");

sortedYears.forEach(year -> yearOptions.add(String.valueOf(year)));

ObservableList<String> yearsList = FXCollections.observableArrayList(yearOptions);

ComboBoxYear.setItems(yearsList);

setCellValueFactories();

filterData();

}

private void setCellValueFactories() {

TableColumnPhoto.setCellValueFactory(cellData -> {

try {

ImageView imageView = new ImageView(cellData.getValue().getPhoto());

imageView.setFitWidth(200);

imageView.setFitHeight(200);

imageView.setPreserveRatio(true);

return new SimpleObjectProperty<ImageView>(imageView);

} catch (IOException e) {

throw new RuntimeException(e);

}

});

TableColumnTitle.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getTitle()));

TableColumnGenre.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getGenre().getTitle()));

TableColumnDescription.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getDescription()));

TableColumnLongs.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getLongs()).asObject());

TableColumnAuthor.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getAuthor().getFirstName() + ' ' + cellData.getValue().getAuthor().getSecondName()));

TableColumnPremier.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getPremier().toString()));

}

@FXML

void TextFieldTextChanged(ActionEvent event) {

filterData();

}

public void TextFieldSearchAction(ActionEvent actionEvent) {

filterData();

}

public void ComboBoxGenreAction(ActionEvent actionEvent) {filterData();}

public void ComboBoxAuthorAction(ActionEvent actionEvent) {filterData();}

public void ComboBoxYearAction(ActionEvent actionEvent) {filterData();}

void filterData() {

List<Performance> performances = performanceService.findAll();

itemsCount = performances.size();

if (!ComboBoxGenre.getSelectionModel().isEmpty()) {

Genre genre = (Genre) ComboBoxGenre.getValue();

if (genre != null && genre.getGenreId() != null && genre.getGenreId() != 0L) {

performances = performances.stream().filter(performance -> performance.getGenre().getGenreId().equals(genre.getGenreId())).collect(Collectors.toList());

}

}

if (!ComboBoxAuthor.getSelectionModel().isEmpty()){

Author author = (Author) ComboBoxAuthor.getValue();

if (author.getAuthorId() != null && author.getAuthorId() != 0L) {

performances = performances.stream().filter(performance -> performance.getAuthor().getAuthorId().equals(author.getAuthorId())).collect(Collectors.toList());

}

}

if (!ComboBoxYear.getSelectionModel().isEmpty()) {

String selectedYear = (String) ComboBoxYear.getValue();

if (selectedYear != null && !selectedYear.equals("Все годa")) {

int year = Integer.parseInt(selectedYear);

performances = performances.stream().filter(performance -> performance.getPremier().getYear() == year).collect(Collectors.toList());

}

}

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

performances = performances.stream().filter(performance -> performance.getTitle().toLowerCase().contains(searchText.toLowerCase())).collect(Collectors.toList());

}

TableViewPerformance.getItems().clear();

for (Performance performance : performances) {

TableViewPerformance.getItems().add(performance);

}

int filteredItemsCount = performances.size();

LabelInfo.setText("Всего записей " + filteredItemsCount + " из " + itemsCount);

}

}

**Страница Расписание**

**Файл интерфейса**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Separator?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.control.ToolBar?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.FlowPane?>

<?import javafx.scene.control.DatePicker?>

<AnchorPane prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.demo.theaterap.controller.ScheduleTableView">

<children>

<BorderPane prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

<top>

<ToolBar prefHeight="40.0" prefWidth="200.0" BorderPane.alignment="CENTER\_RIGHT">

<items>

<Separator halignment="LEFT" maxWidth="1.7976931348623157E308" prefHeight="30.0" prefWidth="225.0" />

<Label fx:id="LabelUser" prefWidth="120.0" text="Label" />

<Button fx:id="BtnAdd" mnemonicParsing="false" onAction="#BtnAddAction" text="Добавить" />

<Button fx:id="BtnUpdate" mnemonicParsing="false" onAction="#BtnUpdateAction" text="Изменить" />

<Button fx:id="BtnDelete" mnemonicParsing="false" onAction="#BtnDeleteAction" text="Удалить" />

<Button fx:id="BtnBack" alignment="CENTER\_RIGHT" mnemonicParsing="false" onAction="#BtnBackAction" text="Назад" textAlignment="RIGHT" />

</items>

</ToolBar>

</top>

<center>

<BorderPane BorderPane.alignment="CENTER">

<top>

<FlowPane minHeight="-Infinity" nodeOrientation="LEFT\_TO\_RIGHT" prefHeight="50.0" prefWidth="200.0" rowValignment="TOP" BorderPane.alignment="CENTER">

<children>

<TextField fx:id="TextFieldSearch" onAction="#TextFieldSearchAction" onInputMethodTextChanged="#TextFieldTextChanged" prefHeight="25.0" prefWidth="262.0" promptText="Введите название для поиска" />

<ComboBox fx:id="ComboBoxGenre" onAction="#ComboBoxGenreAction" prefWidth="150.0" promptText="жанр спектакля" />

<ComboBox fx:id="ComboBoxAuthor" onAction="#ComboBoxAuthorAction" prefWidth="150.0" promptText="автор" />

<DatePicker fx:id="DatePick" onAction="#DatePickerAction" prefWidth="150.0" promptText="дата" />

</children>

</FlowPane>

</top>

<center>

<TableView fx:id="TableViewSchedule" fixedCellSize="200.0" prefHeight="200.0" prefWidth="200.0" tableMenuButtonVisible="true" BorderPane.alignment="CENTER">

<columns>

<TableColumn id="TableColumnPhoto" fx:id="TableColumnPhoto" minWidth="200.0" prefWidth="200.0" resizable="false" text="Фото" />

<TableColumn id="TableColumnTitle" fx:id="TableColumnTitle" maxWidth="1.7976931348623157E308" prefWidth="105.0" text="Название" />

<TableColumn id="TableColumnGenre" fx:id="TableColumnGenre" minWidth="20" prefWidth="67.0" text="Жанр" />

<TableColumn fx:id="TableColumnCost" prefWidth="77.0" text="Цена"/>

<TableColumn fx:id="TableColumnDate" maxWidth="1.7976931348623157E308" prefWidth="98.0" text="Дата и время" />

</columns>

</TableView>

</center>

</BorderPane>

</center>

<bottom>

<Label fx:id="LabelInfo" text="Label" BorderPane.alignment="CENTER\_LEFT" />

</bottom>

</BorderPane>

</children>

</AnchorPane>

**Код Контроллера для страницы**

package ru.demo.theaterap.controller;

import javafx.beans.property.SimpleObjectProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.image.ImageView;

import javafx.scene.input.InputMethodEvent;

import javafx.stage.Modality;

import javafx.stage.Stage;

import ru.demo.theaterap.TheatreAp;

import ru.demo.theaterap.model.\*;

import ru.demo.theaterap.service.AuthorService;

import ru.demo.theaterap.service.GenreService;

import ru.demo.theaterap.service.ScheduleService;

import ru.demo.theaterap.service.TicketService;

import ru.demo.theaterap.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.\*;

import java.util.stream.Collectors;

import static ru.demo.theaterap.util.Manager.ShowConfirmPopup;

public class ScheduleTableView implements Initializable {

@FXML

private Label LabelUser;

@FXML

private Button BtnAdd;

@FXML

private Button BtnUpdate;

@FXML

private Button BtnDelete;

@FXML

private Button BtnBack;

@FXML

private TextField TextFieldSearch;

@FXML

private ComboBox ComboBoxGenre;

@FXML

private ComboBox ComboBoxAuthor;

@FXML

private TableView TableViewSchedule;

@FXML

private TableColumn<Schedule, ImageView> TableColumnPhoto;

@FXML

private TableColumn<Schedule, String> TableColumnTitle;

@FXML

private TableColumn<Schedule, String> TableColumnGenre;

@FXML

private TableColumn<Schedule, String> TableColumnDate;

@FXML

private TableColumn<Schedule, String> TableColumnCost;

@FXML

private DatePicker DatePick;

@FXML

private Label LabelInfo;

@FXML

private ScheduleService scheduleService = new ScheduleService();

@FXML

private TicketService ticketService = new TicketService();

private int itemsCount;

public void BtnAddAction(ActionEvent actionEvent) {

Schedule schedule = (Schedule) TableViewSchedule.getSelectionModel().getSelectedItem();

Manager.currentSchedule = null;

ShowEditScheduleWindow();

filterData();

}

public void BtnUpdateAction(ActionEvent actionEvent) {

Schedule schedule = (Schedule) TableViewSchedule.getSelectionModel().getSelectedItem();

if (schedule != null){

Manager.currentSchedule = schedule;

ShowEditScheduleWindow();

filterData();

}

else

Manager.showErrorMessageBox("Не выбран спектакль для редактирования");

}

public void BtnDeleteAction(ActionEvent actionEvent) {

Schedule schedule = (Schedule) TableViewSchedule.getSelectionModel().getSelectedItem();

List<Ticket> tickets = TicketService.findAll();

if (schedule.getScheduleId() != null && tickets.stream().anyMatch(ticket -> ticket.getScheduleId().getScheduleId().equals(schedule.getScheduleId())))

{

Manager.showErrorMessageBox("Ошибка целостности данных, у данного товара есть зависимые заказы");

return;

}

Optional<ButtonType> result = ShowConfirmPopup();

if (result.get() == ButtonType.OK) {

scheduleService.delete(schedule);

filterData();

}

}

public void BtnBackAction(ActionEvent actionEvent) {

Stage stage = (Stage) BtnBack.getScene().getWindow();

stage.close();

}

public void TextFieldSearchAction(ActionEvent actionEvent) {filterData();}

public void ComboBoxGenreAction(ActionEvent actionEvent) {filterData();}

public void ComboBoxAuthorAction(ActionEvent actionEvent) {filterData();}

public void TextFieldTextChanged(InputMethodEvent inputMethodEvent) {filterData();}

public void DatePickerAction(ActionEvent actionEvent) {filterData();}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

initController();

}

public void initController() {

LabelUser.setText(Manager.currentUser.getFirstName());

List<Genre> genreList = GenreService.findAll();

genreList.add(0, new Genre(0L, "Все жанры"));

ObservableList<Genre> categories = FXCollections.observableArrayList(genreList);

ComboBoxGenre.setItems(categories);

List<Author> authorList = AuthorService.findAll();

authorList.add(0, new Author(0L, "Все ", "авторы"));

ObservableList<Author> authors = FXCollections.observableArrayList(authorList);

ComboBoxAuthor.setItems(authors);

setCellValueFactories();

DatePick.setValue(LocalDate.now());

filterData();

}

private void setCellValueFactories() {

TableColumnPhoto.setCellValueFactory(cellData -> {

try {

ImageView imageView = new ImageView(cellData.getValue().getPerformanceId().getPhoto());

imageView.setFitWidth(200);

imageView.setFitHeight(200);

imageView.setPreserveRatio(true);

return new SimpleObjectProperty<ImageView>(imageView);

} catch (IOException e) {

throw new RuntimeException(e);

}

});

TableColumnTitle.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getPerformanceId().getTitle()));

TableColumnGenre.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getPerformanceId().getGenre().getTitle()));

TableColumnCost.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getPrice() + "руб."));

TableColumnDate.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getDateStart().toString()));

}

void ShowEditScheduleWindow() {

Stage newWindow = new Stage();

FXMLLoader fxmlLoader = new FXMLLoader(TheatreAp.class.getResource("schedule-edit-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

} catch (IOException e) {

throw new RuntimeException(e);

}

newWindow.setTitle("Изменить данные");

newWindow.initOwner(Manager.secondStage);

newWindow.initModality(Modality.WINDOW\_MODAL);

newWindow.setScene(scene);

Manager.currentStage = newWindow;

newWindow.showAndWait();

Manager.currentStage = null;

filterData();}

void filterData() {

List<Schedule> schedules = ScheduleService.findAll();

itemsCount = schedules.size();

if (!ComboBoxGenre.getSelectionModel().isEmpty()) {

Genre genre = (Genre) ComboBoxGenre.getValue();

if (genre != null && genre.getGenreId() != null && genre.getGenreId() != 0L) {

schedules = schedules.stream().filter(schedule -> schedule.getPerformanceId().getGenre().getGenreId().equals(genre.getGenreId())).collect(Collectors.toList());

}

}

if (!ComboBoxAuthor.getSelectionModel().isEmpty()){

Author author = (Author) ComboBoxAuthor.getValue();

if (author.getAuthorId() != null && author.getAuthorId() != 0L) {

schedules = schedules.stream().filter(schedule -> schedule.getPerformanceId().getAuthor().getAuthorId().equals(author.getAuthorId())).collect(Collectors.toList());

}

}

if (!DatePick.getControlCssMetaData().isEmpty()) {

LocalDate selectedDate = DatePick.getValue();

if (selectedDate != null) {

DateTimeFormatter formatter1 = DateTimeFormatter.ofPattern("yyyy/MM/dd");

schedules = schedules.stream().filter(schedule -> schedule.getDateStart().toLocalDate().equals(selectedDate)).collect(Collectors.toList());

}

}

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

schedules = schedules.stream().filter(schedule -> schedule.getPerformanceId().getTitle().toLowerCase().contains(searchText.toLowerCase())).collect(Collectors.toList());

}

TableViewSchedule.getItems().clear();

for (Schedule schedule : schedules) {

TableViewSchedule.getItems().add(schedule);

}

int filteredItemsCount = schedules.size();

LabelInfo.setText("Всего записей " + filteredItemsCount + " из " + itemsCount);

}

}

**Страница Редактирования Расписания**

**Файл интерфейса**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ButtonBar?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.DatePicker?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.ColumnConstraints?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.RowConstraints?>

<AnchorPane prefHeight="447.0" prefWidth="680.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.demo.theaterap.controller.ScheduleEditView">

<children>

<BorderPane layoutX="170.0" layoutY="47.0" prefHeight="200.0" prefWidth="637.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

<bottom>

<ButtonBar prefHeight="40.0" prefWidth="621.0" BorderPane.alignment="CENTER">

<buttons>

<Button fx:id="BtnSave" defaultButton="true" mnemonicParsing="false" onAction="#BtnSaveAction" text="Сохранить" />

<Button fx:id="BtnCancel" cancelButton="true" mnemonicParsing="false" onAction="#BtnCancelAction" text="Отмена" />

</buttons>

<padding>

<Insets right="20.0" />

</padding>

</ButtonBar>

</bottom>

<top>

<GridPane BorderPane.alignment="CENTER">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="200.0" minWidth="10.0" prefWidth="115.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="680.0" minWidth="10.0" prefWidth="485.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="25.0" prefHeight="25.0" />

<RowConstraints maxHeight="112.0" minHeight="25.0" prefHeight="32.0" vgrow="ALWAYS" />

<RowConstraints maxHeight="118.0" minHeight="25.0" prefHeight="30.0" />

<RowConstraints maxHeight="115.0" minHeight="19.0" prefHeight="32.0" />

<RowConstraints maxHeight="246.0" minHeight="27.0" prefHeight="34.0" />

<RowConstraints maxHeight="192.0" minHeight="25.0" prefHeight="185.0" />

<RowConstraints maxHeight="104.0" minHeight="10.0" prefHeight="69.0" />

</rowConstraints>

<children>

<Label maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" text="Спектакль" GridPane.rowIndex="1" />

<TextField fx:id="TextFieldCost" GridPane.columnIndex="1" GridPane.rowIndex="2">

<GridPane.margin>

<Insets right="2.0" top="5.0" />

</GridPane.margin></TextField>

<Label maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" prefHeight="24.0" prefWidth="126.0" text="стоимость" GridPane.rowIndex="2" />

<Label maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" text="дата" GridPane.rowIndex="3" />

<DatePicker fx:id="DatePick" prefHeight="25.0" prefWidth="263.0" GridPane.columnIndex="1" GridPane.rowIndex="3" />

<ComboBox fx:id="ComboBoxPerformance" prefHeight="25.0" prefWidth="261.0" GridPane.columnIndex="1" GridPane.rowIndex="1" />

<Label maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" prefHeight="24.0" prefWidth="126.0" text="номер зала" GridPane.rowIndex="4" />

<ComboBox fx:id="ComboBoxHall" prefHeight="25.0" prefWidth="261.0" GridPane.columnIndex="1" GridPane.rowIndex="4" />

</children>

</GridPane>

</top>

</BorderPane>

</children>

</AnchorPane>

**Код Контроллера для страницы**

package ru.demo.theaterap.controller;

import javafx.collections.FXCollections;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.\*;

import javafx.stage.Stage;

import ru.demo.theaterap.model.Hall;

import ru.demo.theaterap.model.Performance;

import ru.demo.theaterap.model.Schedule;

import ru.demo.theaterap.service.HallService;

import ru.demo.theaterap.service.PerformanceService;

import ru.demo.theaterap.service.ScheduleService;

import ru.demo.theaterap.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.time.LocalTime;

import java.util.ResourceBundle;

import static ru.demo.theaterap.util.Manager.MessageBox;

import static ru.demo.theaterap.util.Manager.currentSchedule;

public class ScheduleEditView implements Initializable {

@FXML

public TextField TextFieldCost;

@FXML

public ComboBox<Performance> ComboBoxPerformance;

public ComboBox<Hall> ComboBoxHall;

@FXML

private Button BtnCancel;

@FXML

private Button BtnSave;

@FXML

private DatePicker DatePick;

private PerformanceService performanceService = new PerformanceService();

private ScheduleService scheduleService = new ScheduleService();

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

ComboBoxPerformance.setItems(FXCollections.observableArrayList(performanceService.findAll()));

ComboBoxHall.setItems(FXCollections.observableArrayList(HallService.findAll()));

if (Manager.currentSchedule != null) {

TextFieldCost.setText(String.valueOf(Manager.currentSchedule.getPrice()));

DatePick.setValue(currentSchedule.getDateStart().toLocalDate());

ComboBoxHall.setValue(currentSchedule.getHall());

ComboBoxPerformance.setValue(currentSchedule.getPerformanceId());

} else {

Manager.currentSchedule = new Schedule();

}

}

public void BtnSaveAction(ActionEvent actionEvent) throws IOException {

String error = checkFields().toString();

if (!error.isEmpty()) {

MessageBox("Ошибка", "Заполните поля", error, Alert.AlertType.ERROR);

return;

}

Manager.currentSchedule.setPerformanceId(ComboBoxPerformance.getValue());

Manager.currentSchedule.setDateStart(DatePick.getValue().atTime(LocalTime.now()));

Manager.currentSchedule.setHall(ComboBoxHall.getValue());

Manager.currentSchedule.setPrice(Double.parseDouble(TextFieldCost.getText().trim()));

if (Manager.currentSchedule.getPerformanceId() == null) {

scheduleService.save(Manager.currentSchedule);

MessageBox("Информация", "", "Данные сохранены успешно", Alert.AlertType.INFORMATION);

} else {

scheduleService.update(Manager.currentSchedule);

MessageBox("Информация", "", "Данные обновлены успешно", Alert.AlertType.INFORMATION);

}

}

public void BtnCancelAction(ActionEvent actionEvent) {

Stage stage = (Stage) BtnCancel.getScene().getWindow();

stage.close();

}

StringBuilder checkFields() {

StringBuilder error = new StringBuilder();

if (TextFieldCost.getText().isEmpty()) {

error.append("Укажите цену\n");

}

try {

if (Double.parseDouble(TextFieldCost.getText().trim()) < 100) {

error.append("Цена не может быть меньше 100\n");

}

} catch (NumberFormatException e) {

error.append("Некорректное значение цены. Введите число.\n");

}

if (ComboBoxPerformance.getValue() == null) {

error.append("Выберите спектакль\n");

}

if (DatePick.getValue() == null) {

error.append("Укажите дату начала спектакля\n");

}

if (ComboBoxHall.getValue() == null) {

error.append("Выберите зал\n");

}

return error;

}

}

**Приложение B**

**Диск с исполняемым кодом программного модуля**