

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

**ОТЧЕТ О ПРОХОЖДЕНИИ ПРАКТИКИ**

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Программная инженерия

Вид практики: производственная практика

Тип практики: Производственно-технологическая

Место прохождения практики: НГУЭУ, кафедра информационных технологий, 630099, г. Новосибирск, ул. Каменская, 52/1

Сроки прохождения практики: с 17 июня 2025 г. по **14 июля 2025 г.**

Выполнил:

Студент гр. ФИ201 14 июля 2025 г. О.В. Зайцев

Руководитель практики

от профильной организации

Заведующий кафедрой ИТ 14 июля 2025 г. А.И. Пестунов

Руководитель практики

от университета

Кандидат технических наук, доцент 14 июля 2025 г. С.Н. Терещенко

Новосибирск 2025



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра **информационных технологий**

**ЗАДАНИЕ НА ПРАКТИКУ**

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Программная инженерия

Вид практики: производственная практика

Тип практики: Производственно-технологическая

Место практики: НГУЭУ, кафедра информационных технологий, 630099, г. Новосибирск, ул. Каменская, 52/1

Сроки прохождения практики: с 17 июня 2025 г. по **14 июля 2025 г.**

Выдано студенту 3 курса ФИ201 группы

Баринову Павлу Александровичу

Индивидуальное задание на практику, содержание, планируемые результаты: разработка веб-приложений на примере реализации информационной системы «Киноактёры». Приложение должно предоставлять пользователю возможность просмотра, фильтрации, поиска данных о спортивных клубах через веб-интерфейс.

Рабочий график (план) проведения практики

|  |  |
| --- | --- |
| Этапы практики | Период |
| Ознакомление с темой | 17.06.25-18.06.25 |
| Составление плана работы, цели, задачи. | 18.06.25-19.06.25 |
| Реализация информационной системы. | 19.06.25-30.06.25 |
| Оформление и подготовка к защите отчета по практике | 1.07.25-14.07.25 |

Руководитель практики

от университета

Старший преподаватель 17 июня 2025 г. Е. Р. Чеглов

Задание согласовано

Руководитель практики от профильной организации

Заведующий кафедрой информационных технологий

А.И. Пестунов 17 июня 2025 г.

Задание получено

Баринов Павел Александрович 17.06.2025



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

**ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ**

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Программная инженерия

Вид практики: производственная практика

Тип практики: Производственно-технологическая

Место практики: НГУЭУ, кафедра информационных технологий, 630099, г. Новосибирск, ул. Каменская, 52/1

Сроки прохождения практики: с 17 июня 2025 г. по **14 июля 2025 г.**

Студент 3 курса, ФИ201 группы

Баринов Павел Александрович

Работал: НГУЭУ

|  |  |  |
| --- | --- | --- |
| Дата | Краткое описание видов работ | Отметка о выполнении |
| 17.06.2025 | Ознакомление с инструкциями организации по правилам противопожарной безопасности, правилам охраны труда, техники безопасности; требованиями по соблюдению санитарно-эпидемиологических правил и гигиенических нормативов; планами эвакуации при возникновении пожара; правилами внутреннего трудового распорядка | ✓ |
| 18.06.2025 | Подборка и анализ научных источников для реализации информационной системы | ✓ |
| 19.06.2025 | Проектирование архитектуры приложения и структуры базы данных | ✓ |
| 25.06.2025 | Разработка серверной части | ✓ |
| 02.07.2025 | Разработка клиентского интерфейса | ✓ |
| 11.07.2025 | Тестирование, демонстрация работы приложения | ✓ |
| 11.07.2025 | Составление и оформление отчета о прохождении практики | ✓ |
| 14.07.2025 | Защита отчета о прохождении практики | ✓ |

Руководитель практики от профильной организации

А. И. Пестунов 14.07.2025 г.

Работы выполнены **в установленные сроки,** содержание практики **соответствует** индивидуальному заданию.

Руководитель практики

от университета

Старший преподаватель 14 июля 2025 г. Е. Р. Чеглов

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc203230223)

[1. Проектирование базы данных 8](#_Toc203230224)

[2. Разработка веб-интерфейса 10](#_Toc203230225)

[2.1 Архитектура приложения 10](#_Toc203230226)

[2.2 Реализация серверной части (Backend-разработка) 11](#_Toc203230227)

[2.3 Реализация клиентской части (Frontend-разработка) 13](#_Toc203230228)

[ЗАКЛЮЧЕНИЕ 16](#_Toc203230229)

[Список Использованных Источников 17](#_Toc203230230)

[ПРИЛОЖЕНИЕ А 18](#_Toc203230231)

# **ВВЕДЕНИЕ**

Веб-разработка играет ключевую роль в автоматизации процессов, обеспечении доступа к данным и улучшении пользовательского опыта. Одной из актуальных задач является создание информационных систем для управления данными, которые позволяют эффективно работать с базой данных через удобный веб-интерфейс.

Целью данной работы является разработка веб-приложения на основе языка программирования Go (Golang), с помощью которого предоставляется пользователю возможность просмотра, фильтрации, поиска данных. Приложение должно быть масштабируемым, производительным и соответствовать требованиям современных стандартов веб-разработки.

Для реализации проекта использованы следующие технологии:

1. язык программирования: Go (Golang). Необходим для разработки серверной части веб-приложения (Backend);
2. база данных: PostgreSQL;
3. клиентская часть (Frontend): HTML, CSS и JavaScript для динамического взаимодействия с сервером.

Основные функциональные возможности приложения включают:

1. просмотр списка всех актеров с предварительной загрузкой данных при старте страницы;
2. поиск актеров по имени или фамилии;
3. фильтрация актеров по гонорару, количеству фильмов в заданном диапазоне и национальности.

В данном отчете будет описан процесс разработки, архитектура приложения, реализация функционала, тестирование и демонстрация работы.

Исходя из поставленной цели, вытекают следующие задачи:

1. разработать структуру базы данных;
2. Реализовать серверную часть на языке Go для создания REST API, обеспечивающего фильтрацию данных.
3. Создать клиентский интерфейс с использованием HTML, CSS и JavaScript, обеспечивающий динамическое взаимодействие с сервером без перезагрузки страницы.
4. Обеспечить интеграцию frontend и backend через HTTP-запросы и формат JSON.

В результате данной работы будет создано веб-приложение, которое демонстрирует весь цикл разработки: от проектирования базы данных до создания интерфейса и тестирования функциональности.

# **Проектирование базы данных**

В начале работы над веб-приложением необходимо заняться проектированием базы данных, в нашем случае на PostgreSQL. В результате проектирования база данных состоит из двух основных таблиц «actors» и «nationalities», более подробная информация предоставлена ниже:

Таблица «actors» содержит информацию об актерах, включая их имя, фамилию, количество фильмов, гонорары и принадлежность к определенной национальности. В данную таблицу входят следующие поля:

* id — уникальный идентификатор актера (первичный ключ);
* first\_name — имя актера (текстовое поле, обязательное для заполнения);
* last\_name — фамилия актера (текстовое поле, обязательное для заполнения);
* nationality\_id — внешний ключ, ссылающийся на таблицу nationalities (обеспечивает связь между актерами и их национальностями);
* film\_count — количество фильмов, в которых снялся актер (целочисленное поле);
* total\_earnings — общий заработок актера в миллионах долларов (число с плавающей точкой).

Таблица «nationalities» содержит информацию о национальностях, к которым могут относиться актеры. Поля у этой таблицы следующие:

* + - id — уникальный идентификатор национальности (первичный ключ).
    - name — название национальности (текстовое поле, уникальное и обязательное для заполнения).

Между таблицами «actors» и «nationalities» установлена связь «один ко многим». Это означает, что одна запись в таблице «nationalities» может быть связана с несколькими записями в таблице «actors». Например, национальность «Американец» может быть указана для нескольких актеров. Связь реализована через внешний ключ «nationality\_id» в таблице «actors», который ссылается на поле «id» в таблице «nationalities».

Далее, после создания таблиц, в базу данных были занесены демонстрационные данные (около 10-15 строк в каждую таблицу). Ознакомиться с результатом проектирования базы данных и внесения в неё данных можно в приложении А.

# **Разработка веб-интерфейса**

## **Архитектура приложения**

Разрабатываемое веб-приложение построено на основе клиент-серверной архитектуры, где серверная часть отвечает за обработку бизнес-логики, взаимодействие с базой данных и предоставление данных через REST API, а клиентская часть реализует удобный пользовательский интерфейс для взаимодействия с этими данными. Такая архитектура обеспечивает четкое разделение обязанностей между компонентами, что делает приложение модульным, гибким и легко расширяемым. Визуальное представление архитектуры представлено в рисунке 1.

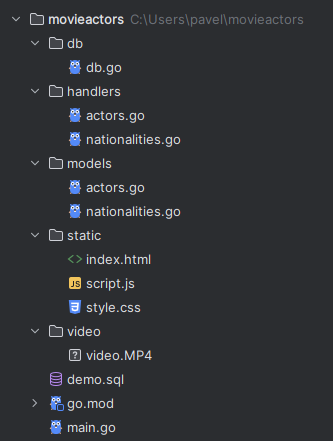


Рисунок 1 – Архитектура приложения

Проект организован в виде нескольких пакетов, каждый из которых выполняет свою конкретную функцию:

main.go: Точка входа в приложение. Здесь настраиваются маршруты HTTP-запросов, подключаются статические файлы (HTML, CSS, JavaScript) и запускается веб-сервер.

demo.sql: файл с SQL-запросами создания необходимых таблиц и добавления в них демонстрационных данных.

db/db.go: Модуль для работы с базой данных PostgreSQL. Содержит функции подключения к базе данных, выполнения миграций и взаимодействия с таблицами.

handlers/actors.go: Обработчики HTTP-запросов для работы с данными об актерах. Реализованы CRUD-операции (создание, чтение, обновление, удаление) и дополнительные функции, такие как поиск и фильтрация.

handlers/nationalities.go: Обработчики для работы с данными о национальностях.

models/actors.go и models/nationalities.go: Определение структур данных, которые соответствуют таблицам в базе данных. Эти модели используются для взаимодействия с базой данных через ORM (GORM).

Статические файлы (static/): HTML, CSS и JavaScript для создания клиентского интерфейса. Эти файлы обеспечивают динамическое взаимодействие с сервером через AJAX-запросы.

Взаимодействие между клиентом и сервером осуществляется посредством HTTP-запросов. Сервер принимает запросы на извлечение, публикацию, вставку и удаление, обрабатывает их и возвращает ответ в формате JSON.

## **Реализация серверной части (Backend-разработка)**

Одно из условий задания было использование в качестве языка для серверной части проекта Go (Golang) — современный компилируемый язык программирования, разработанный корпорацией Google. Он отличается высокой производительностью, встроенной поддержкой параллелизма и развитой стандартной библиотекой, включая пакет net/http для работы с веб-серверами и database/sql для взаимодействия с реляционными базами данных.

Для подключения к базе данных PostgreSQL был использован внешний драйвер github.com/lib/pq, реализующий интерфейс sql/driver и обеспечивающий надёжную работу с PostgreSQL через database/sql.

Подключение к PostgreSQL осуществляется один раз при запуске сервера, с использованием следующей строки подключения:

sql.Open("postgres", "host=localhost port=5432 user=admin password=admin dbname=movieactors sslmode=disable")

Объект \*sql.DB сохраняется и передаётся в функции-обработчики для выполнения SQL-запросов. Такой подход обеспечивает более явный контроль над SQL-запросами и повышает прозрачность логики.

Для обработки HTTP-запросов используется пакет net/http. Каждый маршрут связан с функцией-обработчиком:

http.HandleFunc("/actors", handlers.GetActors)

http.HandleFunc("/nationalities", handlers.GetAllNationalities)

Функции принимают стандартные параметры Go:

* http.ResponseWriter — для отправки ответа клиенту;
* http.Request — для получения данных из запроса (в том числе query-параметров).

В маршруте /actors реализована возможность динамической фильтрации по нескольким параметрам: имя (поиск по имени и фамилии), диапазон гонораров (total\_earnings), диапазон количества фильмов (film\_count), национальность.

Фильтры получаются из query-параметров:

search := r.URL.Query().Get("search")

earningsMin := r.URL.Query().Get("earnings\_min")

filmsMax := r.URL.Query().Get("films\_max")

Далее формируется SQL-запрос. В качестве основы используется запрос с LEFT JOIN, чтобы получить не только данные актёров, но и название национальности по внешнему ключу:

query := `

SELECT a.id, a.first\_name, a.last\_name, a.nationality\_id,

a.film\_count, a.total\_earnings, n.name AS nationality

FROM actors a

LEFT JOIN nationalities n ON a.nationality\_id = n.id

WHERE 1=1

`

Выражение WHERE 1=1 используется как техническая заготовка, позволяющая удобно добавлять фильтры через AND, не проверяя первый ли это фильтр.

Фильтры добавляются простым соединением строк. Такой подход обеспечивает гибкость, но с точки зрения безопасности является менее надёжным. В производственных системах рекомендуется использовать параметризованные запросы с передачей значений отдельно от SQL-текста, однако в рамках данного учебного проекта такой уровень защиты не был реализован, так как акцент делался на изучении базовых принципов обработки запросов и фильтрации данных.

После выполнения SQL-запроса данные считываются в структуру Actor и сериализуются в JSON с помощью стандартного пакета encoding/json:

json.NewEncoder(w).Encode(actors)

Таким образом, frontend получает JSON-объект, который используется для динамического отображения карточек актёров на веб-странице.

## **Реализация клиентской части (Frontend-разработка)**

Клиентская часть проекта была реализована с использованием стандартных веб-технологий: HTML, CSS и JavaScript. В работе не использовались фреймворки, что позволило сосредоточиться на ручной верстке, стилизации и логике взаимодействия с сервером.

Главная задача фронтенда — предоставить пользователю удобный интерфейс для просмотра списка актёров и работы с фильтрами. Страница загружается как одностраничное приложение (скриншот вы можете увидеть на рисунке 2), и сразу после открытия происходит автоматическая загрузка всех актёров из базы данных через асинхронный запрос к серверу. Таким образом, при первом посещении экрана не возникает пустоты — пользователь сразу видит информацию,.

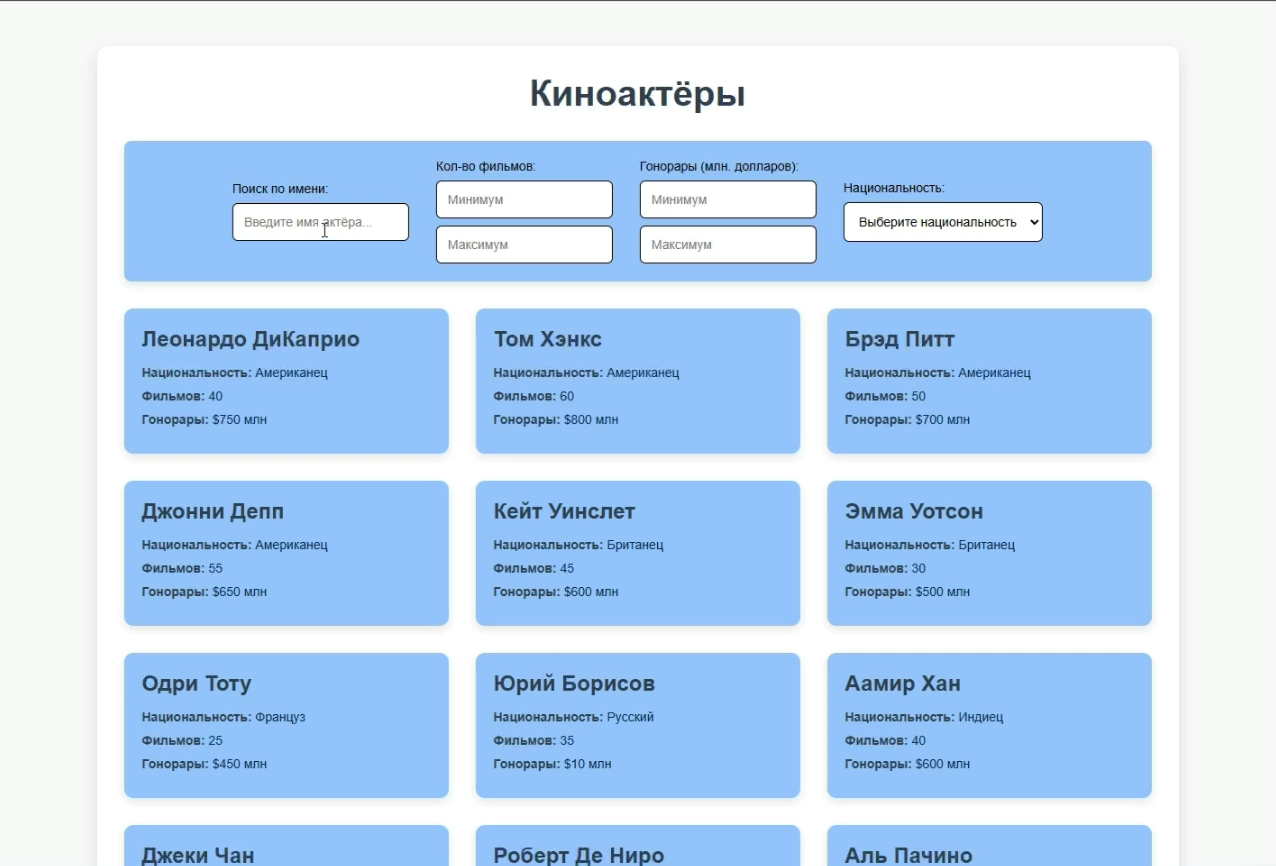


Рисунок 2 – веб-интерфейс, первое посещение экрана.

Интерфейс включает в себя несколько блоков. В верхней части страницы размещён заголовок, под которым находится панель фильтров. Пользователь может ввести текст для поиска по имени, задать диапазон гонораров, указать количество фильмов и выбрать национальность из выпадающего списка. Все фильтры работают в реальном времени: при любом изменении значения автоматически отправляется запрос на сервер, и список актёров обновляется.

Фильтрация реализована так, чтобы можно было сочетать любые параметры. Например, пользователь может одновременно указать имя, диапазон гонорара и количество фильмов. Это позволяет гибко находить нужную информацию. Также реализована защита от пустых фильтров — при их отсутствии сервер возвращает всех актёров.

Отображение данных реализовано в виде карточек. Каждая карточка содержит информацию об актёре: его имя, национальность, количество фильмов и сумма гонораров. Сами карточки выстраиваются в сетку, адаптируются под ширину экрана, а при наведении на них визуально «подпрыгивают» за счёт лёгкой анимации — это придаёт странице динамичность и интерактивность.

Стилизация интерфейса была выполнена вручную. Использованы современные CSS-инструменты: гибкая сетка (Grid и Flexbox), тени, закругления, адаптивные отступы и плавные переходы. Интерфейс адаптирован под разные устройства: на мобильных фильтры и карточки перестраиваются в колонку, чтобы обеспечить удобство восприятия.

Асинхронная работа с сервером реализована через JavaScript. При изменении фильтра или загрузке страницы выполняется запрос к backend'у, в ответ приходит JSON с данными, которые обрабатываются и вставляются в DOM без перезагрузки страницы. Отдельно также загружается список доступных национальностей для выпадающего списка.

Таким образом, клиентская часть получилась лёгкой, наглядной и функциональной. Интерфейс интуитивно понятен, работает быстро, а фильтры позволяют удобно управлять отображаемыми данными. Отказ от сложных библиотек позволил сделать реализацию прозрачной и легко дорабатываемой.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения проекта была успешно реализована полнофункциональная веб-система для отображения и фильтрации информации об актёрах. Работа охватывала все основные этапы разработки: от проектирования базы данных и серверной логики до создания пользовательского интерфейса.

Серверная часть проекта разработана на языке Go, что обеспечило высокую производительность, простоту кода и быструю обработку запросов. Благодаря прямому взаимодействию с базой данных через SQL-запросы удалось реализовать гибкую фильтрацию данных по нескольким параметрам, таким как имя, национальность, гонорары и количество фильмов.

Фронтенд разработан с нуля, с использованием HTML, CSS и JavaScript. Интерфейс получился интуитивно понятным и отзывчивым. Особое внимание было уделено визуальному оформлению: карточки актёров анимированы, фильтры аккуратно выровнены, элементы адаптируются под разные размеры экранов. Вся фильтрация и поиск работают без перезагрузки страницы, что делает использование приложения удобным и современным.

Отдельным плюсом можно отметить разделение кода на логические блоки (модели, обработчики, статические файлы), что способствует читаемости и удобству поддержки проекта в будущем. Структура проекта легко расширяема: при необходимости можно добавить сортировку, редактирование данных или регистрацию пользователей.

Проект успешно решает поставленные задачи и демонстрирует уверенное владение инструментами веб-разработки как на стороне сервера, так и клиента. Полученный опыт может быть использован при разработке более сложных и масштабных систем, а также является хорошей основой для изучения фреймворков и технологий следующего уровня.

Репозитория проекта опубликована на GitHub.

**Список Использованных Источников**

1. The Go Programming Language: Documentation [Электронный ресурс] // Documentation – официальная документация языка программирования Go. — Режим доступа: <https://golang.org/doc/>(дата обращения: 25.06.2025). — Текст: электронный.
2. PostgreSQL: Документация по установке и настройке СУБД [Электронный ресурс] // PostgreSQL Official Documentation – информация о настройке локального сервера PostgreSQL, создании баз данных и пользователей, а также выполнении SQL-скриптов. — Режим доступа: <https://www.postgresql.org/docs/>(дата обращения: 20.06.2025). — Текст: электронный.
3. HTML, CSS и JavaScript: MDN Web Docs [Электронный ресурс] // MDN Web Docs – справочные материалы по разработке клиентской части интерфейса с использованием стандартных технологий веб-разработки. — Режим доступа: <https://developer.mozilla.org/ru/>(дата обращения: 02.07.2025). — Текст: электронный.
4. JavaScript: Современный учебник по разработке клиентской части веб-приложений [Электронный ресурс] // JavaScript.info – подробное описание возможностей JavaScript, AJAX-запросов и динамического обновления DOM-дерева. — Режим доступа: <https://javascript.info/>(дата обращения: 05.07.2025). — Текст: электронный.

# **ПРИЛОЖЕНИЕ А**

Спроектированная база данных:

-- Создание таблицы "Национальности"  
CREATE TABLE nationalities (  
id SERIAL PRIMARY KEY,  
name VARCHAR(100) NOT NULL  
);  
  
-- Создание таблицы "Актеры"  
CREATE TABLE actors (  
id SERIAL PRIMARY KEY,  
first\_name VARCHAR(100) NOT NULL,  
last\_name VARCHAR(100) NOT NULL,  
nationality\_id INTEGER REFERENCES nationalities(id),  
film\_count INTEGER NOT NULL,  
total\_earnings DECIMAL(10, 2) NOT NULL  
);  
  
-- Добавление данных в таблицу "Национальности"  
INSERT INTO nationalities (name) VALUES  
('Американец'),  
('Британец'),  
('Француз'),  
('Русский'),  
('Индиец'),  
('Китаец'),  
('Итальянец'),  
('Немец'),  
('Испанец'),  
('Бразилец'),  
('Канадец'),  
('Казах');  
  
-- Добавление данных в таблицу "Актеры"  
INSERT INTO actors (first\_name, last\_name, nationality\_id, film\_count, total\_earnings) VALUES  
('Леонардо', 'ДиКаприо', 1, 40, 750.00),  
('Том', 'Хэнкс', 1, 60, 800.00),  
('Брэд', 'Питт', 1, 50, 700.00),  
('Джонни', 'Депп', 1, 55, 650.00),  
('Кейт', 'Уинслет', 2, 45, 600.00),  
('Эмма', 'Уотсон', 2, 30, 500.00),  
('Одри', 'Тоту', 3, 25, 450.00),  
('Юрий', 'Борисов', 4, 35, 10.00),  
('Аамир', 'Хан', 5, 40, 600.00),  
('Джеки', 'Чан', 6, 50, 700.00),  
('Роберт', 'Де Ниро', 1, 45, 750.00),  
('Аль', 'Пачино', 1, 50, 700.00),  
('Джулия', 'Робертс', 1, 40, 650.00),  
('Родриго', 'Санторо', 10, 35, 15.00),  
('Николас', 'Кейдж', 1, 55, 700.00);