Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий
Высшая школа Программной инженерии

КУРСОВОЙ ПРОЕКТ

Разработка многопользовательской автоматизированной системы управления организацией. Объект автоматизации – салон музыкальных записей

Выполнил

Студент группы

3530904/00104 Почернин В. С.

Преподаватель Гасанова И. А.

08 декабря 2022 г.

Санкт-Петербург
2022г.

Оглавление

Задание

НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА (КУРСОВОЙ РАБОТЫ)

студенту группы 3530904/00104 Почернин Владислав Сергеевич

(номер группы) *(фамилия, имя, отчество)*

1. Тема проекта (работы) *Разработка многопользовательской автоматизированной системы управления организацией.*

Объект автоматизации - *салон музыкальных записей. Задание № 10*

2. Срок сдачи студентом законченного проекта (работы) 25.12.2022

3. Исходные данные к проекту (работе)

Магазин, продающий музыкальные записи. Клиенты создают заказы, пишут отзывы, читают рекомендации. Сотрудники обрабатывают заказы, пишут рекомендации, следят за работой. Администратор управляет работой магазина.

4.Содержание пояснительной записки (перечень подлежащих разработке вопросов: введение, основная часть (раскрывается структура основной части), заключение, список использованных источников, приложения).

Введение. Анализ предметной области. Проектирование схемы данных. Реализация базы данных в среде SQL Server. Разработка представлений и хранимых процедур. Разработка клиентского приложения. Тестирование. Заключение. Список использованных источников.

Примерный объем пояснительной записки *15-20* страниц машинописного текста

5. Перечень графического материала (с указанием обязательных чертежей и плакатов) *не предоставляется*

6. Консультанты

7. Дата получения задания: «07» __сентября__ 2022 г.

Руководитель Гасанова И.А.

(подпись) *(инициалы, фамилия)*

Задание принял к исполнению Почернин В.С.

(подпись) *(инициалы, фамилия)*

11.09.2022

(дата)

Анализ предметной области

Группы пользователей разрабатываемой информационной системы (ИС)

№ пп	Наименование пользователя
1	Администратор
2	Сотрудник
3	Клиент

Функции групп пользователей

№ пп	Выполняемая функция	Входные данные	Выходные данные	Функции, которые должны быть реализованы в ИС
Администратор				
1	Просмотр всех музыкальных записей.		Таблица **records;**	Вывод на экран администратора всех музыкальных записей.
2	Просмотр всех заказов.		Таблица **orders;**	Вывод на экран администратора всех заказов.
3	Просмотр всех заказов клиента.	email;	Таблица **orders;**	Вывод на экран администратора всех заказов конкретного клиента.
4	Просмотр конкретного заказа.	order_id;	Таблица **orders;** Таблица **orders_records;**	Вывод на экран администратора конкретного заказа.
5	Просмотр всех клиентов.		Таблица **clients;**	Вывод на экран администратора всех клиентов.

6	Просмотр всех отзывов.		Таблица **reviews;**	Вывод на экран администратора всех отзывов.
7	Просмотр всех рекомендаций.		Таблица **recommendations;**	Вывод на экран администратора всех рекомендаций.
8	Просмотр всех сотрудников.		Таблица **employees;**	Вывод на экран администратора всех сотрудников.
9	Добавление музыкальной записи.	title; author; price; count;	Таблица **records;**	Добавление администратором музыкальной записи.
10	Регистрация сотрудника в системе.	first_name; last_name; salary; login; password;	Таблица **logins**; Таблица **employees**;	Добавление администратором сотрудника.
11	Изменение данных сотрудника.	employee_id; first_name; last_name; salary; login; password;	Таблица **logins**; Таблица **employees;**	Изменение администратором данных сотрудника.
12	Изменение данных музыкальной записи.	record_id; title; author; price; count;	Таблица **records;**	Изменение администратором данных музыкальной записи.

13	Удаление отзыва.	review_id;	Таблица **reviews;**	Удаление администратором отзыва.
14	Удаление музыкальной записи.	record_id;	Таблица **records;**	Удаление администратором музыкальной записи.
15	Просмотр всех статусов		Таблица **statuses;**	Вывод на экран администратора всех статусов;
16	Изменение своих данных авторизации	login; password;	Таблица **logins;**	Изменение администратором своих данных авторизации.
Сотрудники				
1	Просмотр всех музыкальных записей.		Таблица **records;**	Вывод на экран сотрудника всех музыкальных записей.
2	Просмотр всех заказов.		Таблица **orders;**	Вывод на экран сотрудника всех заказов.
3	Просмотр всех заказов клиента.	email;	Таблица **orders;**	Вывод на экран сотрудника всех заказов конкретного клиента.
4	Просмотр конкретного заказа.	order_id;	Таблица **orders;** Таблица **orders_records;**	Вывод на экран сотрудника конкретного заказа.
5	Просмотр всех клиентов.		Таблица **clients;**	Вывод на экран сотрудника всех клиентов.
6	Просмотр всех отзывов		Таблица **reviews;**	Вывод на экран сотрудника всех отзывов.

7	Просмотр всех рекомендаций.		Таблица **recommendations;**	Вывод на экран сотрудника всех рекомендаций.
8	Добавление музыкальной записи.	title; author; price; count;	Таблица **records;**	Добавление сотрудником музыкальной записи.
9	Добавление рекомендации для клиента.	record_id (столько, сколько музыкальных записей нужно в подборку); client_id; text;	Таблица **recommendations;** Таблица **recommendations_records;**	Добавление сотрудником рекомендации.
10	Добавление музыкальных записей в рекомендацию.	recommendation_id; record_id (столько, сколько музыкальных записей нужно в подборку);	Таблица **recommendations_records;**	Добавление сотрудником дополнительных музыкальных записей в рекомендацию.
11	Удаление музыкальных записей из рекомендации.	recommendation_id; record_id (столько, сколько музыкальных записей нужно удалить из подборки);	Таблица **recommendations_records;**	Удаление сотрудником музыкальных записей из рекомендации.
12	Изменение текста рекомендации.	recommendation_id; text;	Таблица **recommendations;**	Изменение сотрудником текста рекомендации.

13	Изменение данных музыкальной записи.	record_id; title; author; price; count;	Таблица **records;**	Изменение сотрудником данных музыкальной записи.
14	Обработка заказа клиента.	order_id; status_id;	Таблица **orders;**	Изменение сотрудником статуса заказа.
15	Удаление отзыва.	review_id;	Таблица **reviews;**	Удаление сотрудником отзыва.
16	Удаление рекомендации.	recommendation_id;	Таблица **recommendations;** Таблица **recommendations_records;**	Удаление сотрудником рекомендации.
17	Удаление музыкальной записи.	record_id;	Таблица **records;**	Удаление сотрудником музыкальной записи.
18	Просмотр всех статусов		Таблица **statuses;**	Вывод на экран сотрудника всех статусов;
Клиенты				
1	Просмотр всех музыкальных записей.		Таблица **records;**	Вывод на экран клиента всех музыкальных записей.
2	Просмотр всех своих рекомендаций.		Таблица **recommendations;** Таблица **recommendations_records;**	Вывод на экран клиента всех его рекомендаций.
3	Просмотр всех своих заказов.		Таблица **orders;**	Вывод на экран клиента всех его заказов.

4	Просмотр своего конкретного заказа.	order_id;	Таблица **orders;** Таблица **orders_records;**	Вывод на экран клиента его конкретного заказа.
5	Просмотр отзывов конкретной музыкальной записи.	record_id;	Таблица **reviews;**	Вывод на экран клиента отзывов конкретной музыкальной записи.
5	Создание заказа.	record_id; count (имеется ввиду количество конкретно этой записи в заказе); И так столько пар (record_id – count), сколько нужно разных записей.	Таблица **orders;** Таблица **orders_records;**	Добавление клиентом заказа.
6	Отмена заказа.	order_id;	Таблица **orders;**	Изменение статуса заказа пользователем.
7	Написание отзыва.	record_id; text;	Таблица **reviews;**	Добавление клиентом отзыва.
8	Регистрация клиента.	first_name; last_name; email; login; password; address; dob; phone;	Таблица **clients;** Таблица **logins;**	Добавление клиентом данных о себе.
9	Изменение своих данных.	first_name; last_name;	Таблица **clients;** Таблица **logins;**	Изменение клиентом данных о себе.

		email; login; password; address; dob; phone;		

Хранимые данные

№ пп	Хранимые данные	Клиенты, которым разрешен доступ	Ограничения по типу и значению
1	музыкальные записи (**records**)	**Администратор**: просмотр, добавление, изменение, удаление; **Сотрудник**: просмотр, добавление, изменение, удаление; **Клиент**: просмотр;	
2	Заказы (**orders**)	**Администратор**: просмотр; **Сотрудник**: просмотр, изменение; **Клиент:** просмотр, изменение;	client_id int FOREIGN KEY REFERENCES clients.client_id; employee_id int FOREIGN KEY REFERENCES employees.employee_id; status_id int FOREIGN KEY REFERENCES statuses.status_id;

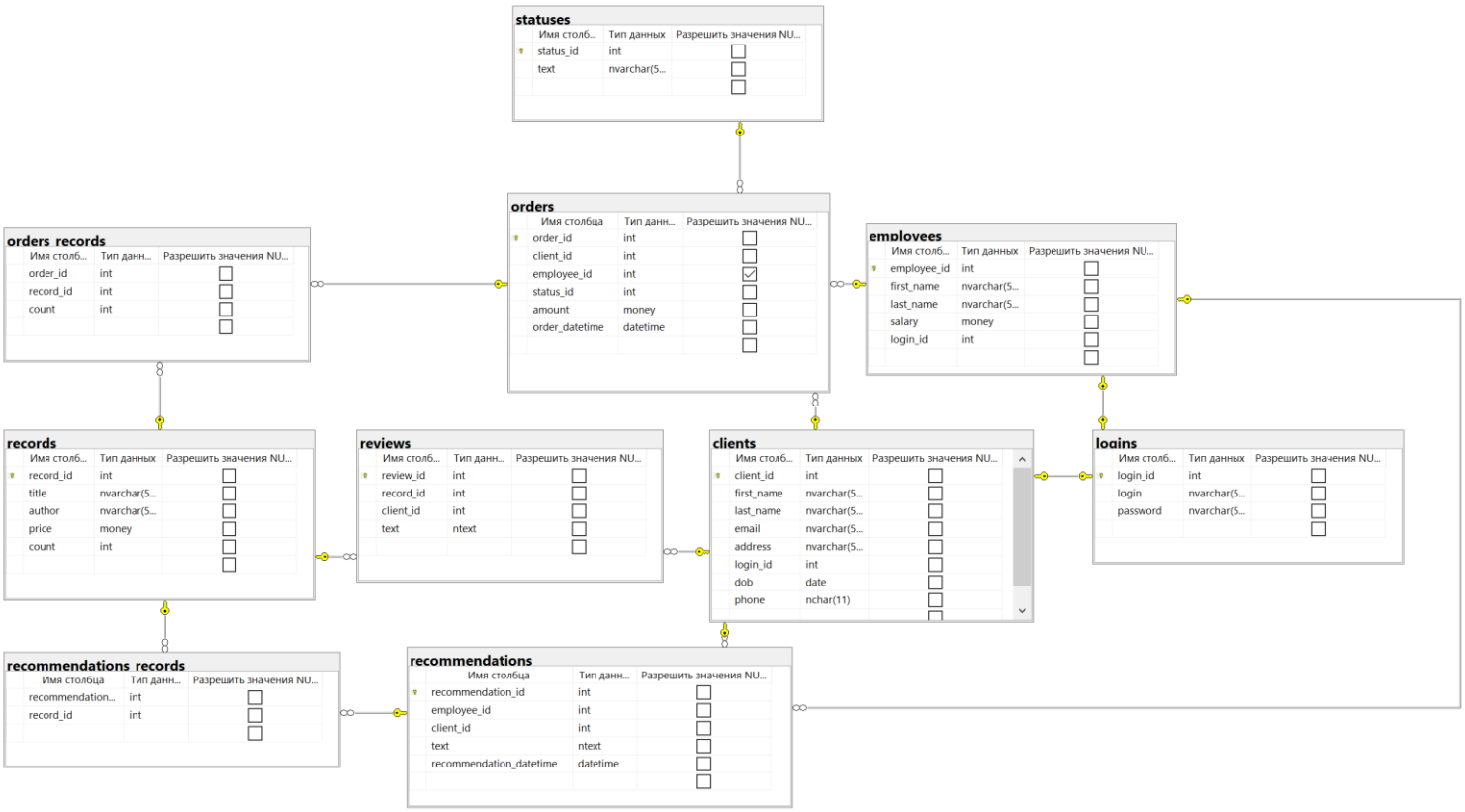
3	Клиенты (**clients**)	**Администратор**: просмотр;
		Сотрудник: просмотр;
		Клиент: добавление;
			email nvarchar(50) UNIQUE;
			login_id int UNIQUE;
			phone nchar(11) CHECK (NOT [phone] like '%[^0-9]%');

4	Отзывы (**reviews**)	**Администратор**: просмотр, удаление**;** **Сотрудник**: просмотр, удаление; **Клиент**: просмотр, добавление;	client_id int FOREIGN KEY REFERENCES clients.client_id; record_id int FOREIGN KEY REFERENCES records.record_id;
5	Рекомендации (**recommendations**)	**Сотрудник**: добавление, изменение, удаление; **Клиент**: просмотр;	client_id int FOREIGN KEY REFERENCES clients.client_id; employee_id FOREIGN KEY REFERENCES employees.employee_id;
6	Сотрудники (**employees**)	**Администратор**: просмотр, добавление, изменение;	login_id int UNIQUE;
7	Данные для входа (**logins**)	**Администратор**: добавление; **Клиент**: добавление;	login nvarchar(50) UNIQUE; login_id int FOREIGN KEY REFERENCES clients.cliend_id; login_id int FOREIGN KEY REFERENCES employees.employee_id;
8	Статусы (**statuses**)	**Администратор**: просмотр; **Сотрудник**: просмотр;	

9	Промежуточная (**orders_records**)	**Администратор**: просмотр; **Сотрудник**: просмотр; **Клиент**: просмотр, добавление, изменение, удаление;	order_id int FOREIGN KEY REFERENCES orders.order_id; record_id int FOREIGN KEY REFERENCES records.record.id;
10	Промежуточная (**recommendations_records**)	**Сотрудник**: добавление, удаление; **Клиента**: просмотр;	recommendation_id int FOREIGN KEY REFERENCES recommendations.recommendation_id; record_id int FOREIGN KEY REFERENCES records.record_id;

Схема с описанием таблиц и связей

Схема БД



Описание таблиц и связей

Название таблицы	Цель таблицы	Связи
statuses	Хранит возможные статусы заказа	Один ко многим statuses.status_id -–orders.status_id;
orders_records	Промежуточная для связи многий ко многим	Многие к одному order_records.order_id – orders.order_id; Многие к одному order_records.record_id – records.record_id;
orders	Хранит данные заказов на покупку музыкальных записей	Многие ко многим с records через orders_records; Многие к одному orders.client_id -–clients.client_id; Многие к одному orders.employee_id – employees.employee_id;
employees	Хранит данные сотрудников	Один к одному employees.login_id – logins.login_id; Один ко многим employees.employee_id – orders.employee_id; Один ко многим employees_employee_id – recommendations.employee_id;

records	Хранит данные музыкальных записей	Один ко многим records.record_id – reviews.record.id; Многие ко многим с recommendations через recommendations_records; Многие ко многим с orders через orders_records;
reviews	Хранит отзывы клиентов на музыкальные записи	Многие к одному reviews.record_id – records.record_id; Многие к одному reviews.client_id – clients.client_id;
clients	Хранит данные клиентов	Один к одному clients.login_id – logins.login_id; Один ко многим clients.client_id – reviews.client_id; Один ко многим clients.client_id – orders.client_id; Один ко многим clients.client_id – recommendations.client_id;
logins	Хранит данные аутентификации	Один к одному logins.login_id – clients.login_id; Один к одному logins.login_id – employees.login_id;
recommendations_records	Промежуточная для связи многий ко многим	Многие к одному recommendations_records.recommendation_id – recommendations.recommendation_id; Многие к одному recommendations_records.record_id – records.record_id;
recommendations	Хранит рекомендации (подборки) музыкальных записей	Многие ко многим с records через recommendations_records; Многие к одному recommendations.employee_id – employees.employee_id; Многие к одному recommendations.client_id – clients.client.id;

Описание каждой таблицы

- Таблица **statuses**

Имя столбца	Тип данных	Разрешить значения NULL
status_id	int	☐
text	nvarchar(50)	☐
		☐

  - **status_id** – номер статуса.
  - **text** – текст статуса.

- Таблица **orders_records**

Имя столбца	Тип данных	Разрешить значения NULL
order_id	int	☐
record_id	int	☐
count	int	☐
		☐

  - **order_id** – номер заказа.
  - **record_id** – номер музыкальной записи.

- Таблица **orders**

Имя столбца	Тип данных	Разрешить значения NULL
order_id	int	☐
client_id	int	☐
employee_id	int	☐
status_id	int	☐
amount	money	☐
order_datetime	datetime	☐
		☐

  - **order_id** – номер заказа.
  - **client_id** – номер клиента, который создал заказ.
  - **employee_id** – номер сотрудника, который обработал заказ.
  - **status_id** – номер статуса заказа.
  - **amount** – общая сумма заказа.
  - **order_datetime** – дата и время заказа.

- Таблица **employees**

Имя столбца	Тип данных	Разрешить значения NULL
order_id	int	☐
client_id	int	☐
employee_id	int	☑
status_id	int	☐
amount	money	☐
order_datetime	datetime	☐
		☐

  - **employee_id –** номер сотрудника.
  - **first_name –** имя сотрудника.

- **last_name –** фамилия сотрудника.
- **salary –** зарплата сотрудника.
- **login_id –** номер данных аутентификации сотрудника.

- Таблица **records**

Имя столбца	Тип данных	Разрешить значения NULL
record_id	int	☐
title	nvarchar(50)	☐
author	nvarchar(50)	☐
price	money	☐
count	int	☐
		☐

  - **record_id** – номер музыкальной записи.
  - **title** – название музыкальной записи.
  - **author** – автор музыкальной записи.
  - **price** – цена музыкальной записи.
  - **count** – количество этих музыкальных записей.

- Таблица **reviews**

Имя столбца	Тип данных	Разрешить значения NULL
review_id	int	☐
record_id	int	☐
client_id	int	☐
text	ntext	☐
		☐

  - **review_id** – номер отзыва.
  - **record_id –** номер музыкальной записи, на которую отзыв.
  - **client_id –** номер клиента, который написал отзыв.
  - **text** – текст отзыва.

- Таблица **clients**

Имя столбца	Тип данных	Разрешить значения NULL
client_id	int	☐
first_name	nvarchar(50)	☐
last_name	nvarchar(50)	☐
email	nvarchar(50)	☐
address	nvarchar(50)	☐
login_id	int	☐
dob	date	☐
phone	nchar(11)	☐
		☐

  - **client_id** – номер клиента.
  - **first_name** – имя клиента.
  - **last_name** – фамилия клиента.
  - **email** – электронная почта клиента.
  - **gender** – пол клиента.
  - **login_id** – номер данных аутентификации клиента.
  - **dob** – дата рождения клиента.

- **phone** – телефонный номер клиента.

- Таблица **logins**

Имя столбца	Тип данных	Разрешить значения NULL
login_id	int	☐
login	nvarchar(50)	☐
password	nvarchar(50)	☐
		☐

  - **login_id** – номер данных аутентификации.
  - **login** – логин.
  - **password** – пароль.

- Таблица **recommendations_records**

Имя столбца	Тип данных	Разрешить значения NULL
recommendation_id	int	☐
record_id	int	☐
		☐

  - **recommendation_id –** номер рекомендации.
  - **record_id** – номер музыкальной записи.

- Таблица **recommendations**

Имя столбца	Тип данных	Разрешить значения NULL
recommendation_id	int	☐
employee_id	int	☐
client_id	int	☐
text	ntext	☐
recommendation_datetime	datetime	☐
		☐

  - **recommendation_id** – номер рекомендации.
  - **employee_id** – номер сотрудника, который создал рекомендацию.
  - **client_id** – номер клиента, для которого рекомендация.
  - **text** – текст рекомендации.
  - **recommendation_datetime –** дата и время создания рекомендации.

Запросы

ВЫВОД

ОДНОТАБЛИЧНЫЕ Представления

- Вывод всех музыкальных записей (таблица **records**);

```
CREATE VIEW showAllRecords AS
	SELECT record_id AS 'Артикул', title AS 'Название', author AS 'Автор', price AS
	'Цена', count AS 'Количество'
	FROM records;
```

	Артикул	Название	Автор	Цена	Количество
1	1	Smoke on the water	Deep Purpule	100,00	63
2	2	Бесполезно	Валентин Стрыкало	90,00	92
3	3	25 к 10	Аквариум	70,22	102
4	4	Hey you	Pink Floyd	101,20	296
5	5	Cosmos2016	Yuki	170,32	11
6	6	Гоша Рубчинский	Слава КПСС	10,00	1

- Вывод всех заказов (таблица **orders**);

```
CREATE VIEW showAllOrders AS
	SELECT order_id AS 'Номер заказа', client_id AS 'Номер клиента', employee_id AS
	'Номер сотрудника', text AS 'Статус заказа',
		amount AS 'Сумма заказа', order_datetime AS 'Дата и время заказа'
	FROM orders, statuses
	WHERE orders.status_id = statuses.status_id;
```

	Номер заказа	Номер клиента	Номер сотрудника	Статус заказа	Сумма заказа	Дата и время заказа
1	1	1	1	Заказ обработан сотрудником	1000,00	2022-11-23 12:25:56.607
2	2	2	2	Заказ отменен сотрудником	1301,10	2022-11-23 12:25:56.610
3	3	1	NULL	Заказ отменен клиентом	520,44	2022-11-23 12:26:05.847
4	4	1	1	Заказ обработан сотрудником	1351,10	2022-12-05 18:53:54.627
5	5	1	1	Заказ обработан сотрудником	303,60	2022-12-05 18:54:17.513
6	6	1	1	Заказ обработан сотрудником	9000,00	2022-12-05 18:54:22.893
7	7	3	1	Заказ обработан сотрудником	4051,20	2022-12-05 19:35:08.150
8	8	3	1	Заказ обработан сотрудником	800,00	2022-12-05 21:50:48.820
9	9	3	NULL	Заказ создан	14044,00	2022-12-05 22:56:41.013

- Вывод всех клиентов (таблица **clients**);

```
CREATE VIEW showAllClients AS
	SELECT client_id AS 'Номер клиента', first_name AS 'Имя', last_name AS 'Фамилия',
email AS 'Электронная почта',
		address AS 'Адрес', dob AS 'Дата рождения', phone AS 'Номер телефона'
	FROM clients;
```

	Номер клиента	Имя	Фамилия	Электронная почта	Адрес	Дата рождения	Номер телефона
1	1	Василий	Пупкин	pupkin@gmail.com	Адрес Пупкина	2000-10-20	79876543210
2	2	Иван	Иванov	ivanov@mail.ru	Адрес Иванова	2000-10-21	70123456789
3	3	Владислав	Почернин	vspochernin@gmail.com	Санкт-Петербург	2008-12-04	12345678901

- Вывод всех отзывов (таблица **reviews**);

```
CREATE VIEW showAllReviews AS
	SELECT review_id AS 'Номер отзыва', reviews.record_id AS 'Артикул', title AS 'Название записи',
		client_id AS 'Номер клиента', text AS 'Текст рекомендации'
	FROM reviews, records
	WHERE reviews.record_id = records.record_id;
```

	Номер отзыва	Артикул	Название записи	Номер клиента	Текст рекомендации
1	1	2	Бесполезно	1	Теперь это моя любимая песня!
2	2	1	Smoke on the water	2	Рок - отстой!
3	3	6	Гоша Рубчинский	3	Крутая

- Вывод всех рекомендаций (таблица **recommendations**);

```
CREATE VIEW showAllRecommendations AS
	SELECT recommendation_id AS 'Номер рекомендации', employee_id AS 'Номер сотрудника', client_id AS 'Номер клиента',
		text AS 'Текст рекомендации', recommendation_datetime AS 'Дата и время рекомендации'
	FROM recommendations;
```

	Номер рекомендации	Номер сотрудника	Номер клиента	Текст рекомендации	Дата и время рекомендации
1	1	1	1	Очень рекомендую послушать!	2022-11-23 12:23:00.473
2	2	2	2	Ненавижу свою работу!	2022-11-23 12:23:00.477

- Вывод всех сотрудников (таблица **employees**);

```
CREATE VIEW showAllEmployees AS
	SELECT employee_id AS 'Номер сотрудника', first_name AS 'Имя', last_name AS 'Фамилия',
		salary AS 'Зарплата'
	FROM employees;
```

	Номер сотрудника	Имя	Фамилия	Зарплата
1	1	Хороший	Продавец	30000,00
2	2	Плохой	Продавец	15000,00

- Вывод всех статусов (таблица **statuses**);

```
CREATE VIEW showAllStatuses AS
	SELECT text AS 'Текст статуса'
	FROM statuses;
```

	Текст статуса
1	Заказ создан
2	Заказ обработан сотрудником
3	Заказ отменен клиентом
4	Заказ отменен сотрудником

ОДНОТАБЛИЧНЫЕ Хранимые процедуры

- Вывод всех заказов конкретного клиента (email) (таблица **orders**);

```
CREATE PROCEDURE showOrdersByClientEmail(@email AS NVARCHAR(50)) AS
	SELECT order_id AS 'Номер заказа', client_id AS 'Номер клиента', employee_id AS 'Номер сотрудника',
		text AS 'Статус заказа', amount AS 'Сумма заказа', order_datetime AS 'Дата и время заказа'
	FROM orders, statuses
	WHERE orders.client_id = (SELECT client_id FROM clients WHERE clients.email = @email)
		AND orders.status_id = statuses.status_id;
```

	Номер заказа	Номер клиента	Номер сотрудника	Статус заказа	Сумма заказа	Дата и время заказа
1	1	1	1	Заказ обработан сотрудником	1000,00	2022-11-23 12:25:56.607
2	3	1	NULL	Заказ отменен клиентом	520,44	2022-11-23 12:26:05.847
3	4	1	1	Заказ обработан сотрудником	1351,10	2022-12-05 18:53:54.627
4	5	1	1	Заказ обработан сотрудником	303,60	2022-12-05 18:54:17.513
5	6	1	1	Заказ обработан сотрудником	9000,00	2022-12-05 18:54:22.893

- Вывод всех заказов конкретного клиента (id клиента) (таблица **orders**);

```
CREATE PROCEDURE showOrdersByClientId(@client_id AS int) AS
	SELECT order_id AS 'Номер заказа', client_id AS 'Номер клиента', employee_id AS 'Номер сотрудника',
		text AS 'Статус заказа', amount AS 'Сумма заказа', order_datetime AS 'Дата и время заказа'
	FROM orders, statuses
	WHERE orders.client_id = @client_id
		AND orders.status_id = statuses.status_id;
```

	Номер заказа	Номер клиента	Номер сотрудника	Статус заказа	Сумма заказа	Дата и время заказа
1	1	1	1	Заказ обработан сотрудником	1000,00	2022-11-23 12:25:56.607
2	3	1	NULL	Заказ отменен клиентом	520,44	2022-11-23 12:26:05.847
3	4	1	1	Заказ обработан сотрудником	1351,10	2022-12-05 18:53:54.627
4	5	1	1	Заказ обработан сотрудником	303,60	2022-12-05 18:54:17.513
5	6	1	1	Заказ обработан сотрудником	9000,00	2022-12-05 18:54:22.893

- Вывод отзывов к конкретной музыкальной записи (id музыкальной записи) (таблица **reviews**);

```
CREATE PROCEDURE showReviewsOfRecord(@record_id AS int) AS
	SELECT review_id AS 'Номер отзыва', client_id AS 'Номер клиента',
reviews.record_id AS 'Артикул', title AS 'Название',
		author AS 'Автор', text AS 'Текст рекомендации'
	FROM reviews, records
	WHERE reviews.record_id = @record_id AND records.record_id = @record_id;
```

	Номер отзыва	Номер клиента	Артикул	Название	Автор	Текст рекомендации
1	1	1	2	Бесполезно	Валентин Стрыкало	Теперь это моя любимая песня!

МНОГОТАБЛИЧНЫЕ Хранимые процедуры

- Вывод конкретного заказа (id заказа) (таблицы **orders**, **orders_records**);

```
CREATE PROCEDURE showOrderById(@order_id AS int) AS
	SELECT orders_records.record_id AS 'Артикул', title AS 'Название', author AS
'Автор', orders_records.count AS 'Количество'
	FROM orders
		JOIN orders_records ON orders_records.order_id = orders.order_id
		JOIN records ON records.record_id = orders_records.record_id
		WHERE orders.order_id = @order_id;
```

	Артикул	Название	Автор	Количество
1	1	Smoke on the water	Deep Purpule	5
2	2	Бесполезно	Валентин Стрыкало	5
3	3	25 к 10	Аквариум	5

- Вывод всех рекомендаций конкретному клиенту (id клиента); (таблицы **recommendations**, **recommendations_records**);

```
CREATE PROCEDURE showRecommendationsOfClient(@client_id AS int) AS
	SELECT recommendation_id AS 'Номер рекомендации', employee_id AS 'Номер
сотрудника', client_id AS 'Номер клиента',
		text AS 'Текст рекомендации', recommendation_datetime AS 'Дата и время
рекомендации'
		FROM recommendations
		WHERE client_id = @client_id;

CREATE PROCEDURE showRecommendationById(@recommendation_id AS int) AS
	SELECT recommendations_records.record_id AS 'Артикул', title AS 'Название', author
AS 'Автор'
		FROM recommendations
		JOIN recommendations_records ON recommendations_records.recommendation_id =
recommendations.recommendation_id
		JOIN records ON records.record_id = recommendations_records.record_id
		WHERE recommendations.recommendation_id = @recommendation_id
		ORDER BY recommendations.recommendation_id;
```

	Номер рекомендации	Номер сотрудника	Номер клиента	Текст рекомендации	Дата и время рекомендации
1	1	1	1	Очень рекомендую послушать!	2022-11-23 12:23:00.473

	Артикул	Название	Автор
1	1	Smoke on the water	Deep Purpule
2	2	Бесполезно	Валентин Стрыкало
3	3	25 к 10	Аквариум

- Вывод топ 10 самых продаваемых треков;

```
CREATE VIEW showMostPurchased AS
	SELECT TOP 10
	records.record_id AS Артикул, title AS Название, author AS Автор, price AS Цена,
SUM(orders_records.count) AS 'Количество продаж'
	FROM records
	JOIN orders_records ON orders_records.record_id = records.record_id
	JOIN orders ON orders.order_id = orders_records.order_id
	WHERE orders.status_id = 2
	GROUP BY records.record_id, title, author, price
	ORDER BY 'Количество продаж' DESC;
```

	Артикул	Название	Автор	Цена	Количество продаж
1	2	Бесполезно	Валентин Стрыкало	90,00	115
2	1	Smoke on the water	Deep Purpule	100,00	54
3	3	25 к 10	Аквариум	70,22	5
4	4	Hey you	Pink Floyd	101,20	4

ДОБАВЛЕНИЕ

ОДНОТАБЛИЧНЫЕ Хранимые процедуры

- Добавление музыкальной записи (title, author, price, count) (таблица **records**);

```
CREATE PROCEDURE addRecord(@title AS NVARCHAR(50), @author AS NVARCHAR(50), @price AS
money, @count AS int) AS
	INSERT INTO records(title, author, price, count) VALUES (@title, @author, @price,
@count);
```

- Добавление рекомендации (id ссотрудника, id клиента, text) (таблица **recommendations**);

```
CREATE PROCEDURE addRecommendation(@employee_id AS int, @client_id AS int, @text as
NTEXT) AS
	INSERT INTO recommendations(employee_id, client_id, text, recommendation_datetime)
VALUES
	(@employee_id, @client_id, @text, GETDATE());
```

- Добавление музыкальной записи в рекомендацию (id рекомендации, id добавляемой музыкальной записи) (таблица **recommendations_records**);

```
CREATE PROCEDURE addRecordToRecommendation(@recommendation_id AS int, @record_id AS int)
AS
	INSERT INTO recommendations_records(recommendation_id, record_id) VALUES
	(@recommendation_id, @record_id);
```

- Добавление отзыва (id музыкальной записи, text) (таблица **reviews**);

```
CREATE PROCEDURE addReview(@record_id AS int, @client_id AS int, @text AS NTEXT) AS
	INSERT INTO reviews(record_id, client_id, text) VALUES (@record_id, @client_id,
@text);
```

- Добавление заказа (id клиента) (таблица **orders**);

```
CREATE PROCEDURE addOrder(@client_id AS int) AS
	INSERT INTO orders(client_id, status_id, amount, order_datetime) VALUES
	(@client_id, 1, 0, GETDATE());
```

МНОГОТАБЛИЧНЫЕ Хранимые процедуры

- Добавление сотрудника (first_name, last_name, salary, login, password) (таблицы **logins**, **employees**);

```
CREATE PROCEDURE addEmployee(@first_name AS NVARCHAR(50), @last_name AS NVARCHAR(50),
	@salary AS money, @login AS NVARCHAR(50), @password AS NVARCHAR(50)) AS
	BEGIN
		BEGIN TRY
			BEGIN TRANSACTION;

			INSERT INTO logins(login, password) VALUES(@login, @password)

			DECLARE @login_id int = @@IDENTITY;

			INSERT INTO employees(first_name, last_name, salary, login_id) VALUES
				(@first_name, @last_name, @salary, @login_id)

			COMMIT TRANSACTION
		END TRY
		BEGIN CATCH
			ROLLBACK TRANSACTION;
			THROW
		END CATCH
	END;
```

- Добавление музыкальной записи в заказ (id заказа, id музыкальной записи, count) (таблицы **orders**, **orders_records**, **records**);

```
CREATE PROCEDURE addRecordToOrder(@order_id AS int, @record_id AS int, @count AS int) AS
	BEGIN
		BEGIN TRY
			BEGIN TRANSACTION;
```

```
                    DECLARE @record_count int = (SELECT records.count FROM records WHERE
records.record_id = @record_id);
                    DECLARE @record_price money = (SELECT records.price FROM records
WHERE records.record_id = @record_id);
                    DECLARE @order_amount money = (SELECT orders.amount FROM orders WHERE
orders.order_id = @order_id);
                    DECLARE @order_status int = (SELECT orders.status_id FROM orders
WHERE orders.order_id = @order_id);

                    IF (@order_status != 1)
                    BEGIN
                           THROW 50000, 'Невозможно добавить записи в завершенный заказ',
1;
                    END

                    IF (@count > @record_count)
                    BEGIN
                           THROW 50000, 'Недостаточно музыкальных записей на складе', 1;
                    END

                    UPDATE records
                    SET
                           records.count = @record_count - @count
                    WHERE
                           records.record_id = @record_id

                    UPDATE orders
                    SET
                           orders.amount = @order_amount + (@record_price * @count)
                    WHERE
                           orders.order_id = @order_id;

                    IF (EXISTS (SELECT * FROM orders_records WHERE
orders_records.order_id = @order_id AND orders_records.record_id = @record_id))
                    BEGIN
                           DECLARE @orders_records_count money = (SELECT
orders_records.count FROM orders_records
                                  WHERE orders_records.order_id = @order_id AND
orders_records.record_id = @record_id);
                           UPDATE orders_records
                           SET
                                  orders_records.count = @orders_records_count + @count
                           WHERE
                                  orders_records.order_id = @order_id AND
orders_records.record_id = @record_id;
                    END
                    ELSE
                    BEGIN
                           INSERT INTO orders_records(order_id, record_id, count) VALUES
(@order_id, @record_id, @count)
                    END

                    COMMIT TRANSACTION
             END TRY
             BEGIN CATCH
                    ROLLBACK TRANSACTION;
                    THROW
             END CATCH
      END;
```

- Добавление клиента (first_name, last_name, email, login, password, address, dob, phone) (таблицы **clients**, **logins**);

```
CREATE PROCEDURE addClient(@first_name AS NVARCHAR(50), @last_name AS NVARCHAR(50),
@email AS NVARCHAR(50), @login AS NVARCHAR(50),
	@password AS NVARCHAR(50), @address AS NVARCHAR(50), @dob AS date, @phone AS
NCHAR(11)) AS
	BEGIN
		BEGIN TRY
			BEGIN TRANSACTION;

			INSERT INTO logins(login, password) VALUES(@login, @password)

			DECLARE @login_id int = @@IDENTITY;

			INSERT INTO clients(first_name, last_name, email, address, login_id,
dob, phone) VALUES
				(@first_name, @last_name, @email, @address, @login_id, @dob,
@phone)

			COMMIT TRANSACTION
		END TRY
		BEGIN CATCH
			ROLLBACK TRANSACTION;
			THROW
		END CATCH
	END;
```

ИЗМЕНЕНИЕ

ОДНОТАБЛИЧНЫЕ Хранимые процедуры

- Изменение данных музыкальной записи (id музыкальной записи, title; author; price; count) (таблица **records**);

```
CREATE PROCEDURE updateRecord(@record_id AS int, @title AS NVARCHAR(50), @author AS
NVARCHAR(50), @price AS MONEY, @count AS int) AS
	UPDATE records
	SET
		records.title = @title,
		records.author = @author,
		records.price = @price,
		records.count = @count
	WHERE
		records.record_id = @record_id;
```

- Изменение данных авторизации администратора (login, password) (таблица **logins**);

```
CREATE PROCEDURE updateAdmin(@login AS NVARCHAR(50), @password AS NVARCHAR(50)) AS
	UPDATE logins
	SET
		logins.login = @login,
		logins.password = @password
	WHERE logins.login_id = 1;
```

- Изменение текста рекомендации (id рекомендации, text);

```
CREATE PROCEDURE updateRecommendationText(@recommendation_id AS int, @text AS NTEXT) AS
	UPDATE recommendations
	SET
		recommendations.text = @text
```

```
	WHERE
		recommendations.recommendation_id = @recommendation_id;
```

- Изменение статуса заказа (id заказа, id статуса) (таблица **orders**);

```
CREATE PROCEDURE updateStatusByClient(@order_id AS int, @status_id AS int) AS
	BEGIN
		BEGIN TRY
			BEGIN TRANSACTION;

			DECLARE @current_status_id int = (SELECT orders.status_id FROM orders
WHERE orders.order_id = @order_id);

			IF (@current_status_id = '3' OR @current_status_id = '4')
			BEGIN
				THROW 50000, 'Заказ был отменен, его статус не изменить', 1;
			END

			IF (@current_status_id = '2')
			BEGIN
				THROW 50000, 'Заказ был обработан, его статус не изменить', 1;
			END

			IF (@status_id = '1')
			BEGIN
				THROW 50000, 'Статус созданного заказа невозможно изменить
вручную', 1;
			END

			IF (@status_id = '3' OR @status_id = '4')
			BEGIN
				DECLARE @cursor CURSOR;
				DECLARE @record_id int;

				SET @cursor = CURSOR FOR SELECT orders_records.record_id FROM
orders_records WHERE orders_records.order_id = @order_id;

				OPEN @cursor
				FETCH NEXT FROM @cursor INTO @record_id;

				WHILE @@FETCH_STATUS = 0
				BEGIN
					DECLARE @current_count int = (SELECT records.count FROM
records WHERE records.record_id = @record_id);

					UPDATE records
					SET
						records.count = @current_count +
						(SELECT orders_records.count FROM orders_records
WHERE orders_records.order_id = @order_id AND orders_records.record_id = @record_id)
					WHERE
						records.record_id = @record_id;

					FETCH NEXT FROM @cursor INTO @record_id
				END;

				CLOSE @cursor;
				DEALLOCATE @cursor;
			END

			UPDATE orders
			SET
```

```
                        orders.status_id = @status_id
                WHERE
                        orders.order_id = @order_id;

                        COMMIT TRANSACTION
        END TRY
        BEGIN CATCH
                        ROLLBACK TRANSACTION;
                        THROW
        END CATCH
END;


CREATE PROCEDURE updateStatusByEmployee(@order_id AS int, @status_id AS int, @employee_id
AS int) AS
        BEGIN
                BEGIN TRY
                        BEGIN TRANSACTION;

                        DECLARE @current_status_id int = (SELECT orders.status_id FROM orders
WHERE orders.order_id = @order_id);

                        IF (@current_status_id = '3' OR @current_status_id = '4')
                        BEGIN
                                THROW 50000, 'Заказ был отменен, его статус не изменить', 1;
                        END

                        IF (@current_status_id = '2')
                        BEGIN
                                THROW 50000, 'Заказ был обработан, его статус не изменить', 1;
                        END

                        IF (@status_id = '1')
                        BEGIN
                                THROW 50000, 'Статус созданного заказа невозможно изменить
вручную', 1;
                        END

                        IF (@status_id = '3' OR @status_id = '4')
                        BEGIN
                                DECLARE @cursor CURSOR;
                                DECLARE @record_id int;

                                SET @cursor = CURSOR FOR SELECT orders_records.record_id FROM
orders_records WHERE orders_records.order_id = @order_id;

                                OPEN @cursor
                                FETCH NEXT FROM @cursor INTO @record_id;

                                WHILE @@FETCH_STATUS = 0
                                BEGIN
                                        DECLARE @current_count int = (SELECT records.count FROM
records WHERE records.record_id = @record_id);

                                        UPDATE records
                                        SET
                                                records.count = @current_count +
                                                (SELECT orders_records.count FROM orders_records
WHERE orders_records.order_id = @order_id AND orders_records.record_id = @record_id)
                                        WHERE
                                                records.record_id = @record_id;

                                        FETCH NEXT FROM @cursor INTO @record_id
                                END;
```

```
                        CLOSE @cursor;
                        DEALLOCATE @cursor;
                END

                UPDATE orders
                SET
                        orders.status_id = @status_id,
                        orders.employee_id = @employee_id
                WHERE
                        orders.order_id = @order_id;

                COMMIT TRANSACTION
        END TRY
        BEGIN CATCH
                ROLLBACK TRANSACTION;
                THROW
        END CATCH
END;
```

МНОГОТАБЛИЧНЫЕ Хранимые процедуры

- Изменение данных сотрудника (id сотрудника, first_name, last_name, salary, login, password) (таблицы **logins**, **employees**);

```
CREATE PROCEDURE updateEmployee(@employee_id AS int, @first_name AS NVARCHAR(50),
@last_name AS NVARCHAR(50),
        @salary AS money, @login AS NVARCHAR(50), @password AS NVARCHAR(50)) AS
        BEGIN
                BEGIN TRY
                        BEGIN TRANSACTION;

                        DECLARE @login_id int = (SELECT employees.login_id FROM employees
WHERE employees.employee_id = @employee_id);

                        UPDATE logins
                        SET
                                logins.login = @login,
                                logins.password = @password
                        WHERE
                                logins.login_id = @login_id;


                        UPDATE employees
                        SET
                                employees.first_name = @first_name,
                                employees.last_name = @last_name,
                                employees.salary = @salary
                        WHERE
                                employees.employee_id = @employee_id;

                        COMMIT TRANSACTION
                END TRY
                BEGIN CATCH
                        ROLLBACK TRANSACTION;
                        THROW
                END CATCH
        END;
```

- Изменение данных клиента (id клиента, first_name, last_name, email, login, password, address, dob, phone) (таблицы **clients**, **logins**);

```
CREATE PROCEDURE updateClient(@client_id AS int, @first_name AS NVARCHAR(50), @last_name
AS NVARCHAR(50), @email AS NVARCHAR(50), @login AS NVARCHAR(50),
	@password AS NVARCHAR(50), @address AS NVARCHAR(50), @dob AS date, @phone AS
NCHAR(11)) AS
	BEGIN
		BEGIN TRY
			BEGIN TRANSACTION;

			DECLARE @login_id int = (SELECT clients.login_id FROM clients WHERE
clients.client_id = @client_id);

			UPDATE logins
			SET
				logins.login = @login,
				logins.password = @password
			WHERE
				logins.login_id = @login_id;

			UPDATE clients
			SET
				clients.first_name = @first_name,
				clients.last_name = @last_name,
				clients.email = @email,
				clients.address = @address,
				clients.dob = @dob,
				clients.phone = @phone
			WHERE
				clients.client_id = @client_id;

			COMMIT TRANSACTION
		END TRY
		BEGIN CATCH
			ROLLBACK TRANSACTION;
			THROW
		END CATCH
	END;
```

УДАЛЕНИЕ

ОДНОТАБЛИЧНЫЕ Хранимые процедуры

- Удаление отзыва (id отзыва) (таблица **reviews**);

```
CREATE PROCEDURE removeReview(@review_id AS int) AS
	DELETE FROM reviews WHERE review_id = @review_id;
```

- Удаление музыкальной записи **(**id музыкальной записи) (таблица **records**);

```
CREATE PROCEDURE removeRecord(@record_id AS INT) AS
	UPDATE records
	SET
		records.count = 0
	WHERE
		records.record_id = @record_id;
```

- Удаление музыкальной записи из рекомендации (id рекомендации, id удаляемой музыкальной записи (таблица **recommendations_records**);

```
CREATE PROCEDURE removeRecordFromRecommendation(@recommendation_id AS int, @record_id AS
int) AS
	DELETE FROM recommendations_records WHERE
	recommendations_records.recommendation_id = @recommendation_id AND
recommendations_records.record_id = @record_id;
```

МНОГОТАБЛИЧНЫЕ Хранимые процедуры

- Удаление рекомендации (id рекомендации) (таблицы **recommendations**, **recommendations_records**);

```
CREATE PROCEDURE removeRecommendation(@recommendation_id AS int) AS
	DELETE FROM recommendations_records WHERE
recommendations_records.recommendation_id = @recommendation_id
	DELETE FROM recommendations WHERE recommendations.recommendation_id =
@recommendation_id;
```

Описание программы

При разработке курсового проекта была использована система управления базами данных (СУБД) компании Microsoft – Microsoft SQL Server.

Клиентское приложение было написано на объектно-ориентированном языке программирования C# на платформе .NET Windows Forms.

Соединение с базой данных осуществлялось посредством встроенных библиотек платформы .NET, а именно с помощью драйвера Microsoft SqlClient Data Provider for SQL Server.

Для шифрования паролей пользователей использовался алгоритм хэширования MD5.

Весь процесс разработки кода клиентского приложения велся в среде разработки Microsoft Visual Studio 2022.

Клиентское приложение

Авторизация

При запуске приложения, пользователя встречает окно авторизации:

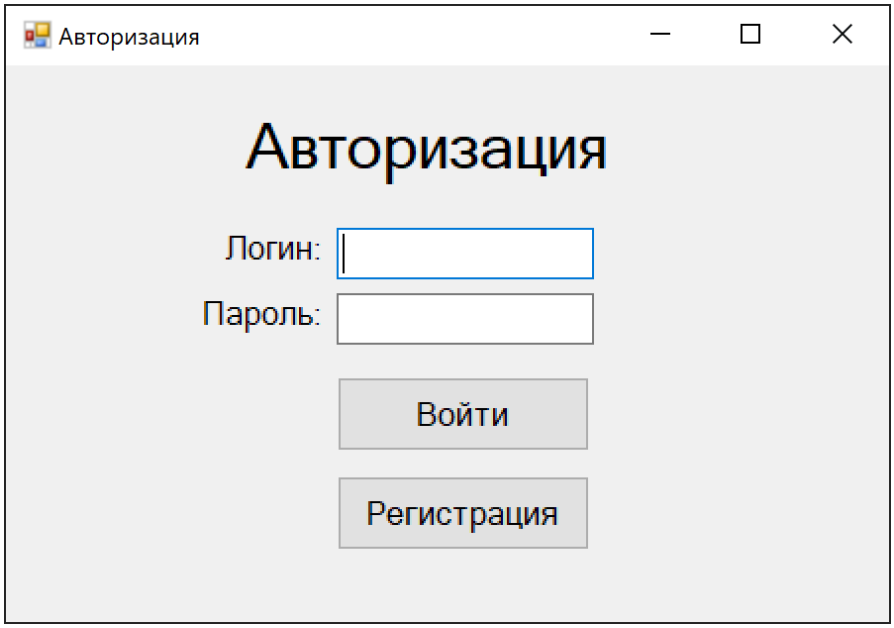


Рисунок 1 Окно авторизации

Если пользователь еще не зарегистрирован, он может сделать это:

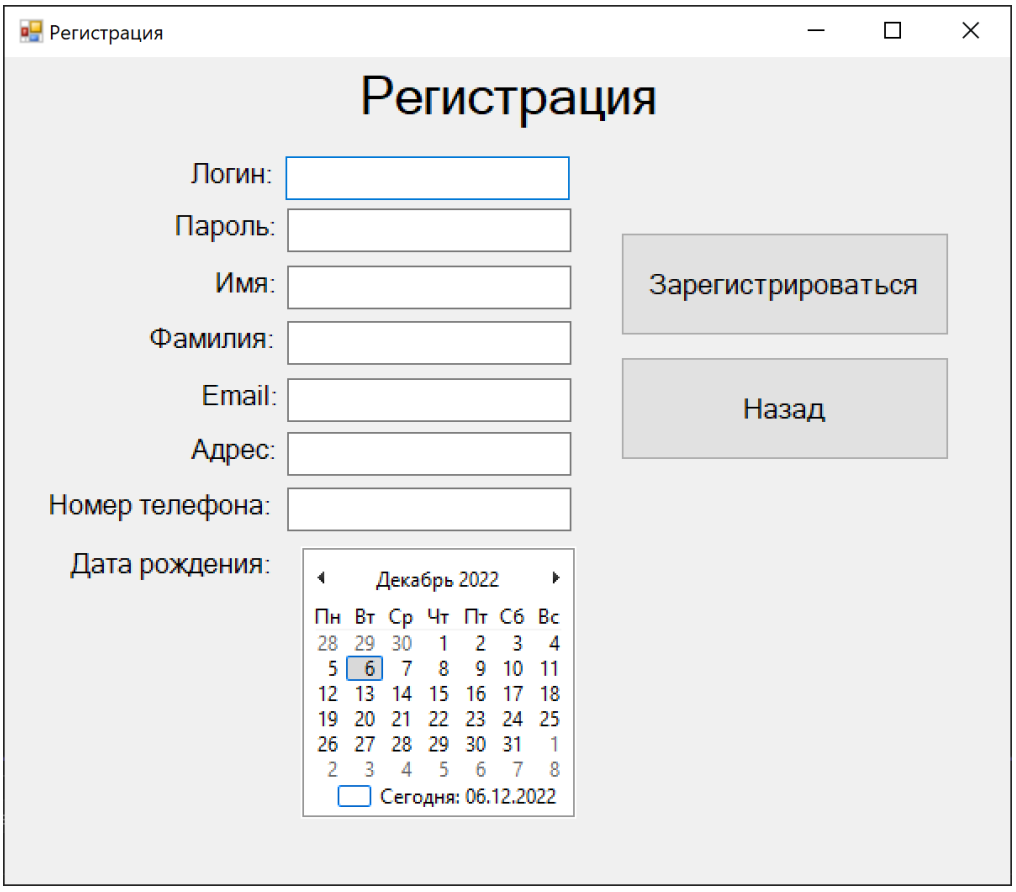


Рисунок 2 Окно регистрации

Интерфейс клиента

Если клиент зайдет в систему, его встретит каталог музыкальных записей:

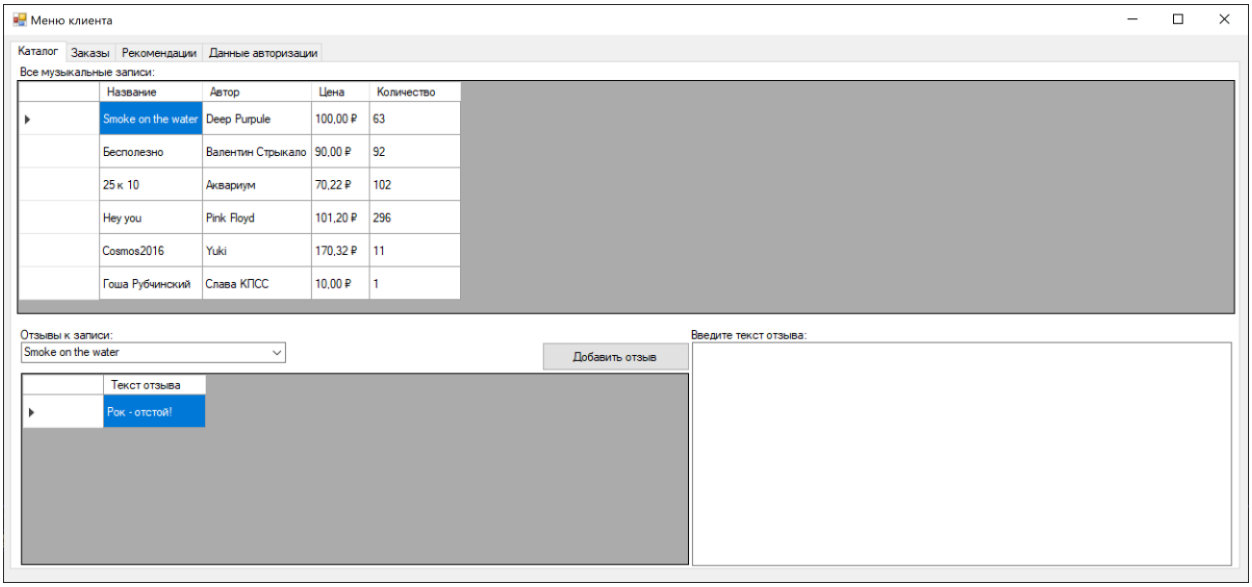


Рисунок 3 Окно каталога музыкальных записей для клиента

Также клиент может управлять своими заказами:

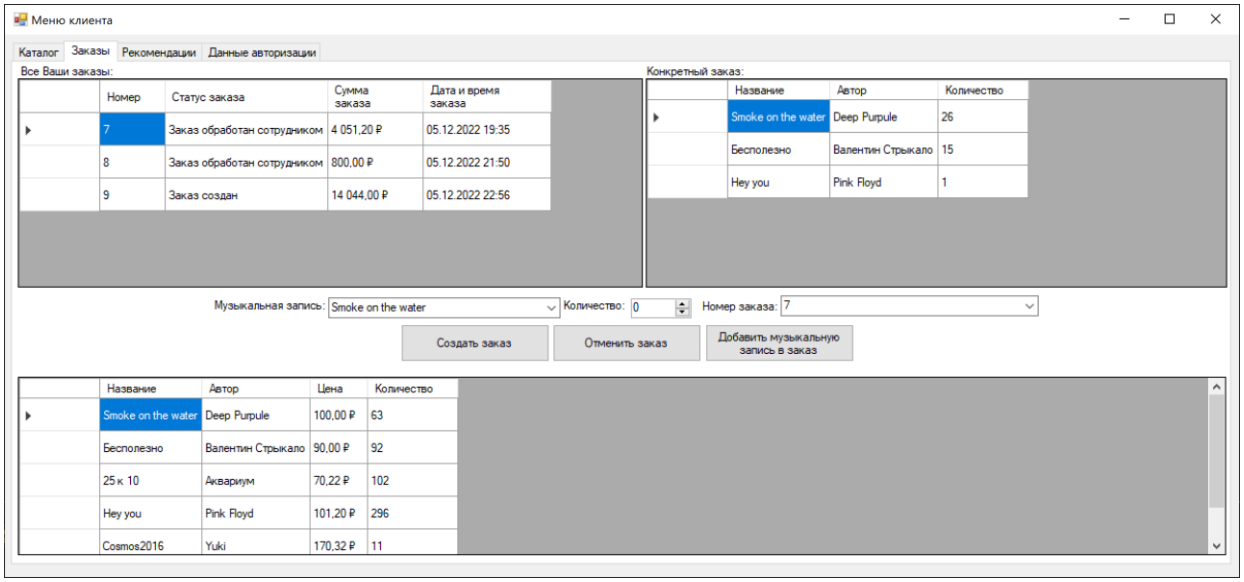


Рисунок 4 Окно заказов для клиента

Смотреть свои рекомендации:

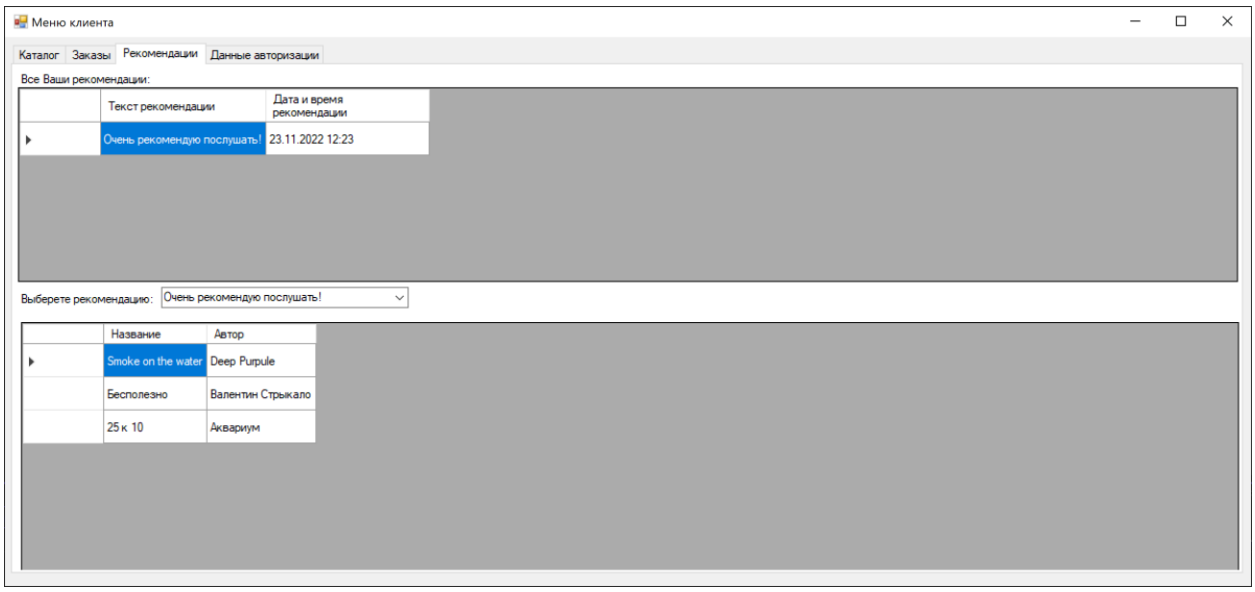


Рисунок 5 Окно рекомендаций для клиента

А также изменить свои данные авторизации:

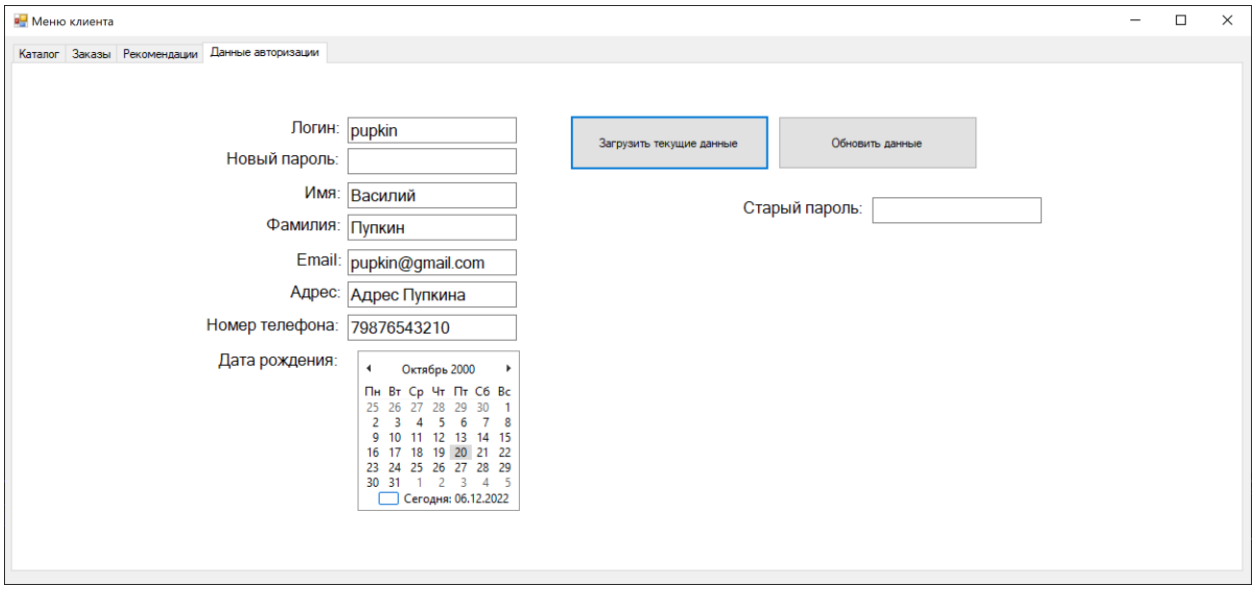


Рисунок 6 Окно изменения данных авторизации для клиента

Интерфейс сотрудника

Сотрудник также начинает работу с музыкальных записей:

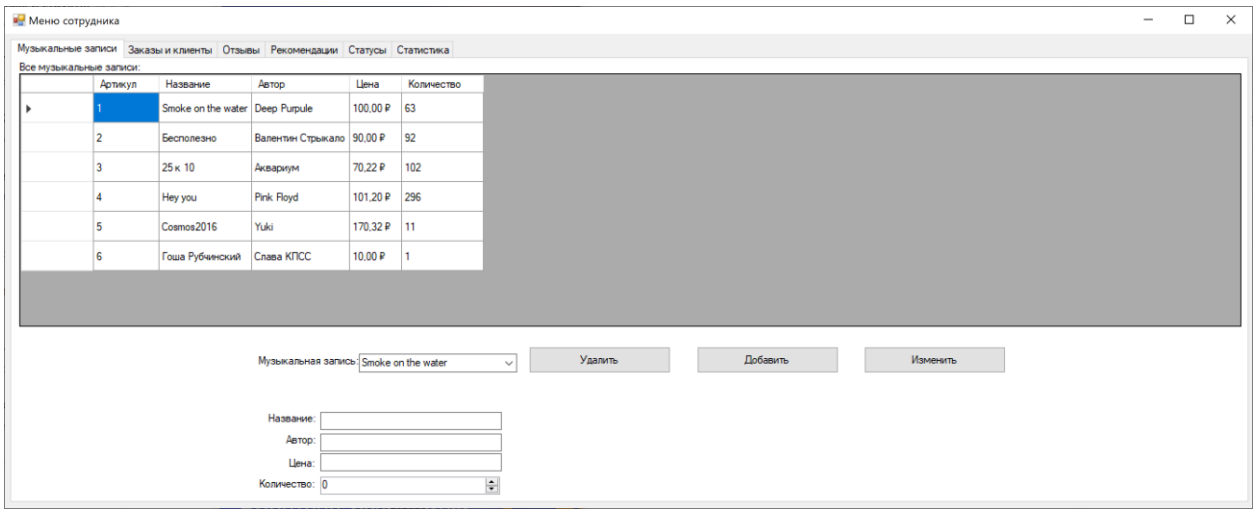


Рисунок 7 Окно музыкальных записей для сотрудника

Также он может управлять заказами:

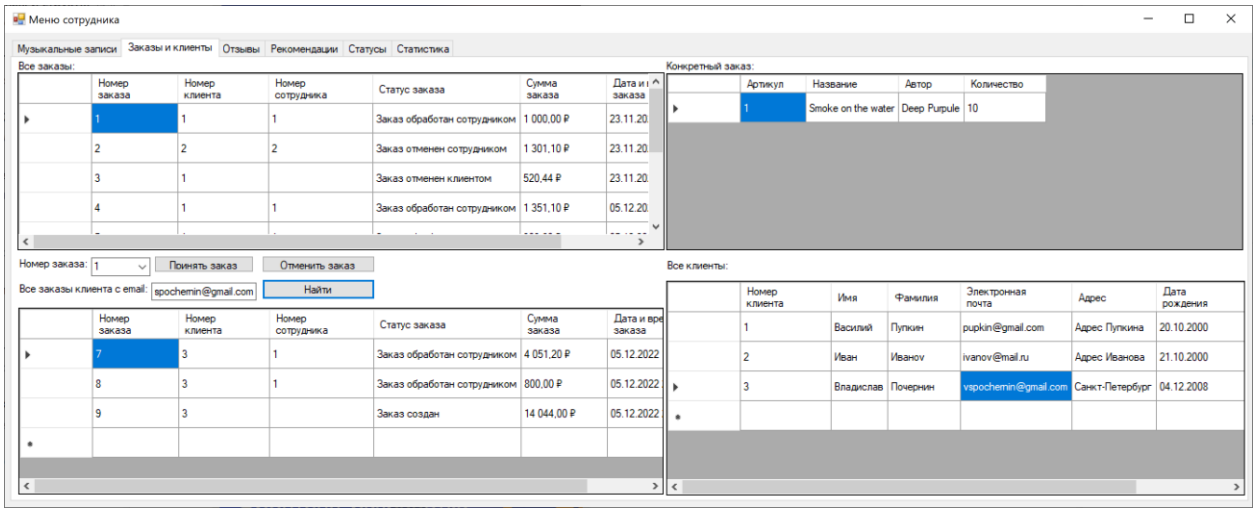


Рисунок 8 Окно заказов и клиентов для сотрудника

Смотреть и удалять отзывы:

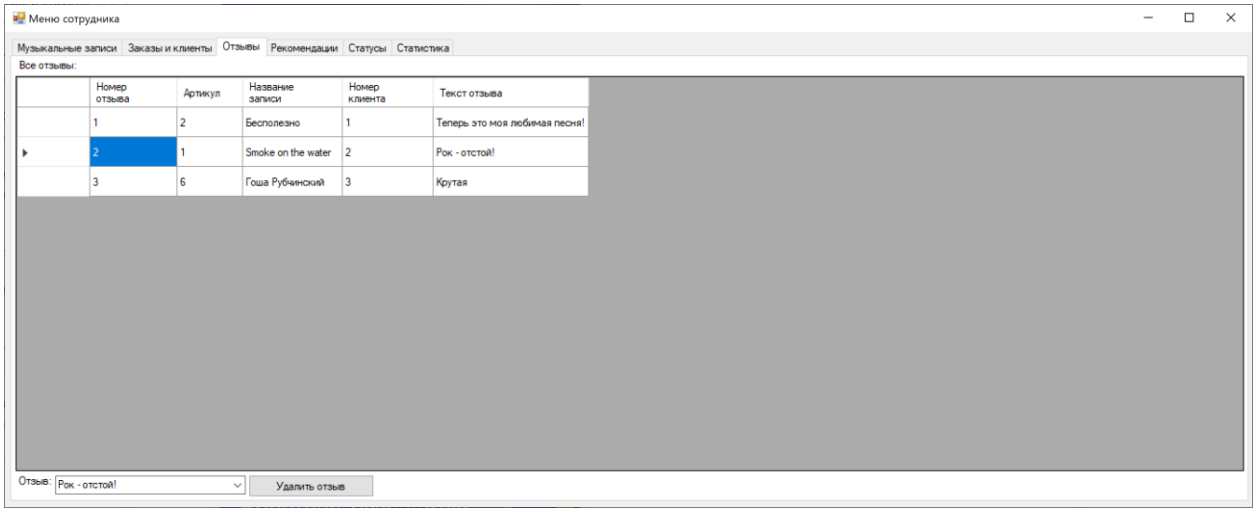


Рисунок 9 Окно отзывов для сотрудника

Писать рекомендации:

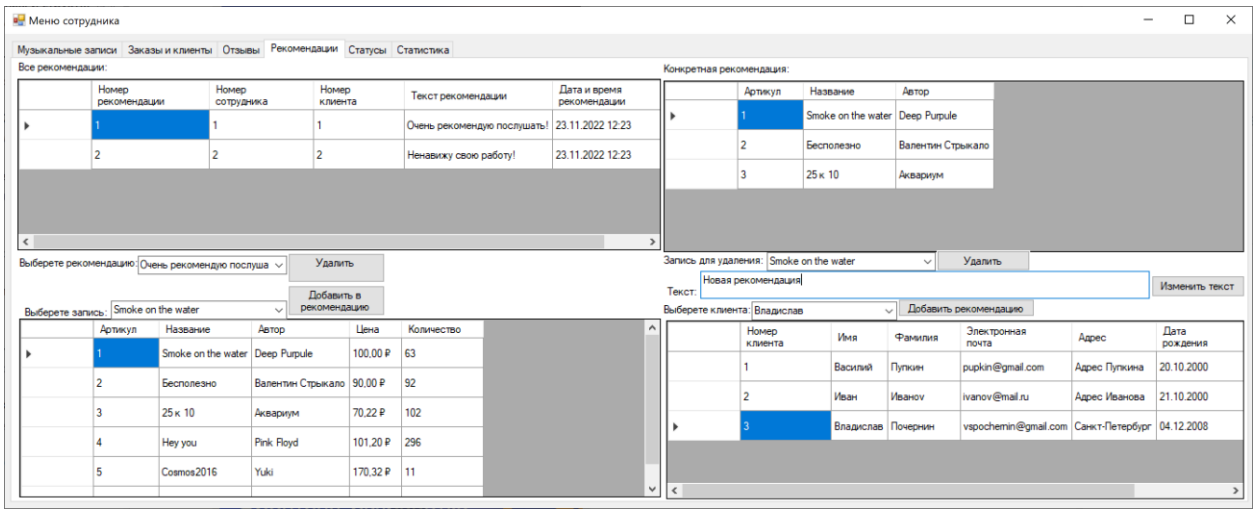


Рисунок 10 Окно рекомендаций для сотрудника

Смотреть статусы:

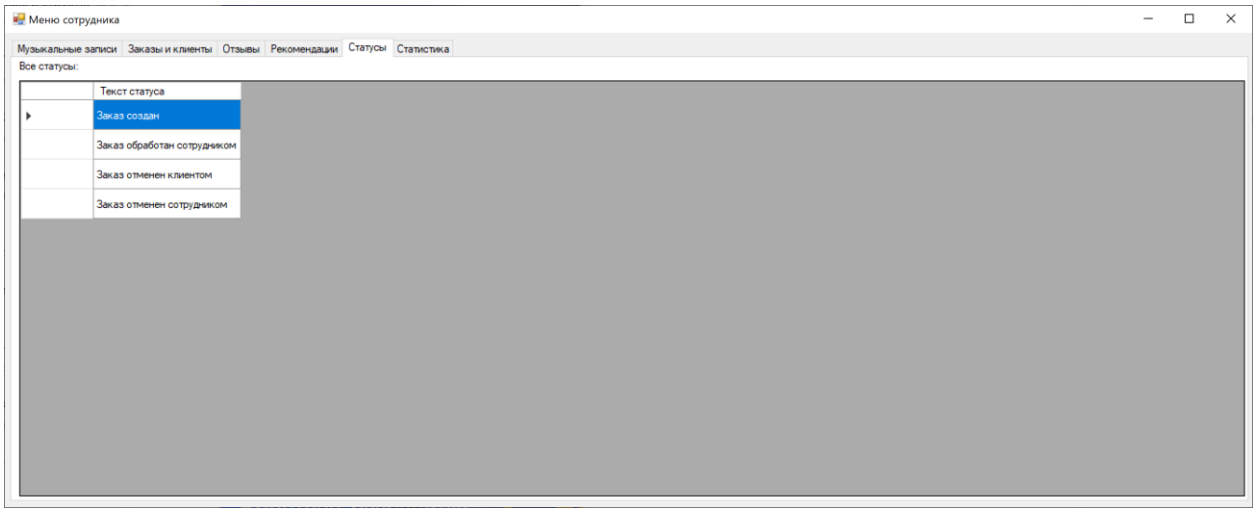


Рисунок 11 Окно статусов для сотрудника

А также наблюдать статистику по проданным песням:

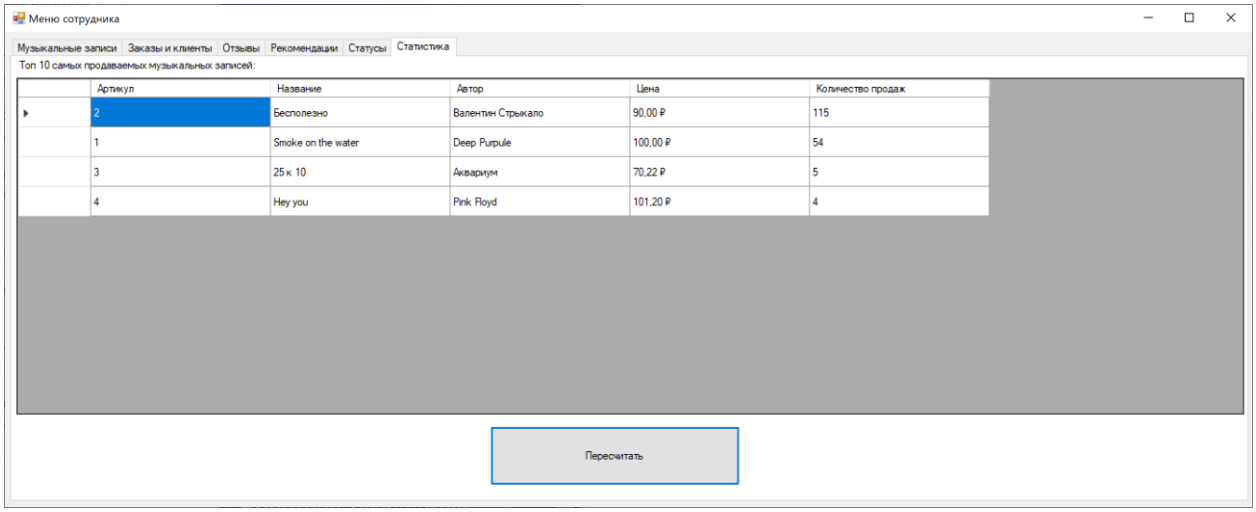


Рисунок 12 Окно статистики для сотрудника

Интерфейс администратора

Интерфейс администратора частично совпадает с интерфейсом сотрудника:

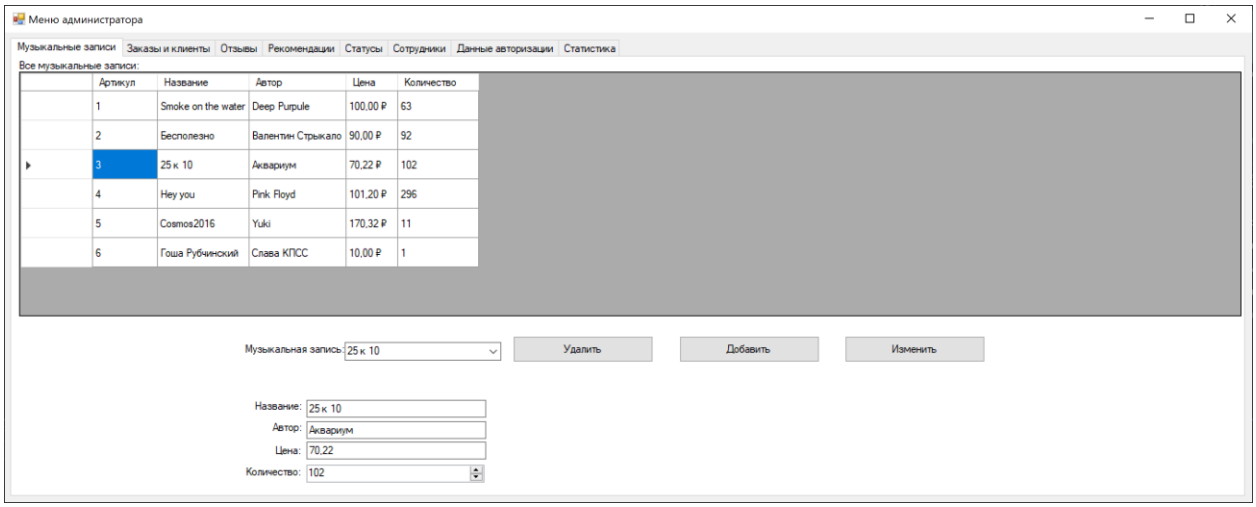


Рисунок 13 Окно музыкальных записей для администратора

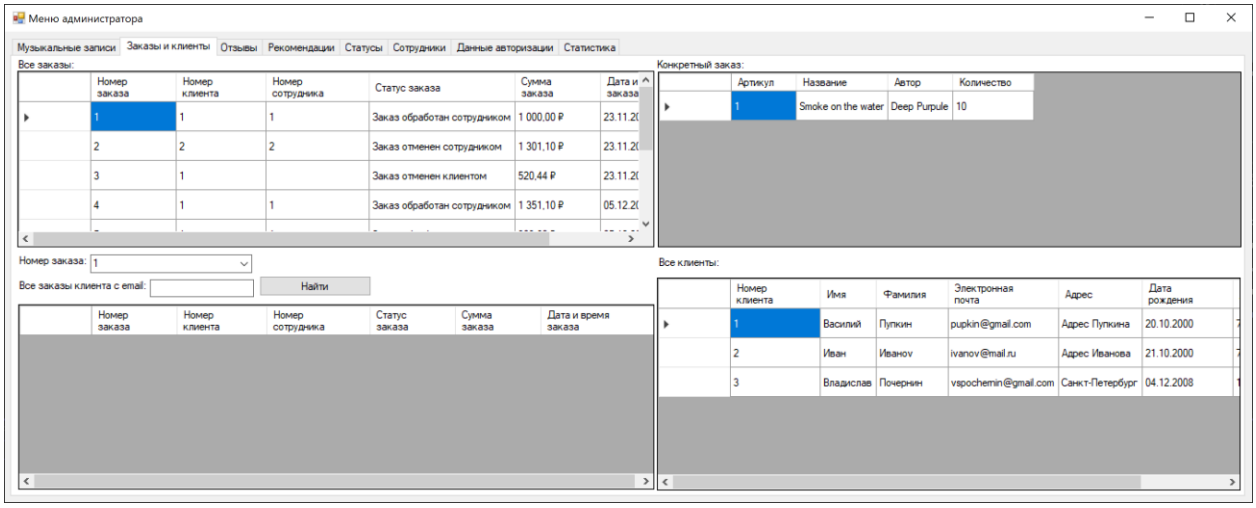


Рисунок 14 Окно заказов и клиентов для администратора

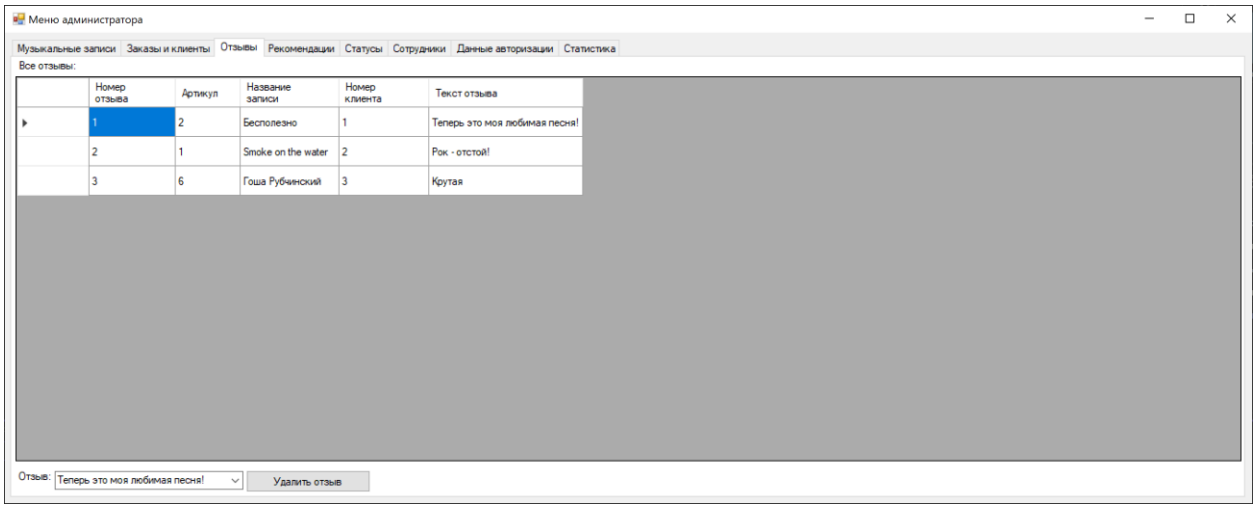


Рисунок 15 Окно отзывов для администратора

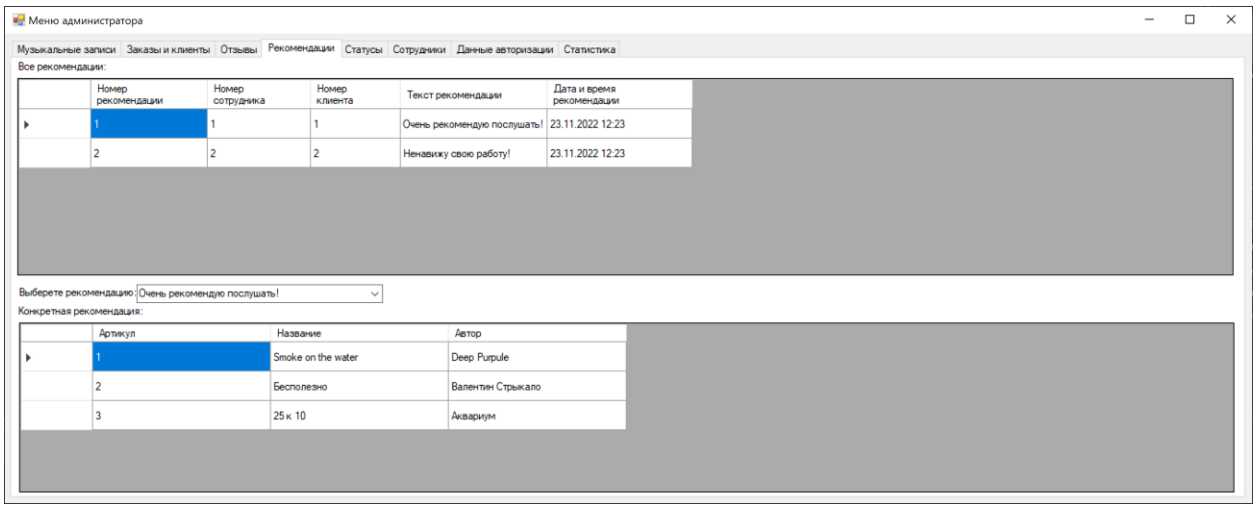


Рисунок 16 Окно рекомендаций для администратора

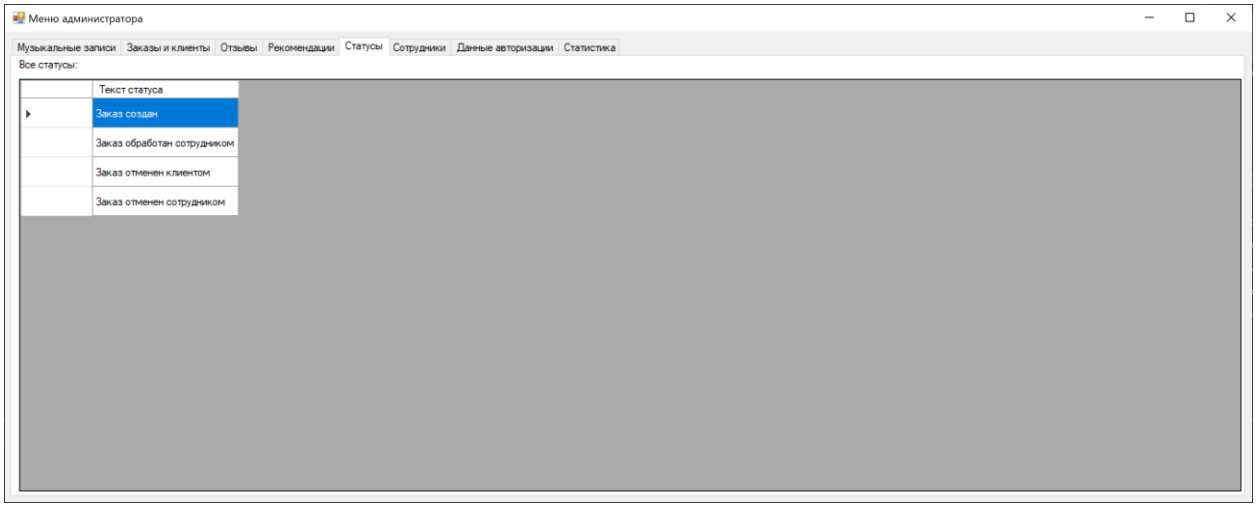


Рисунок 17 Окно статусов для администратора

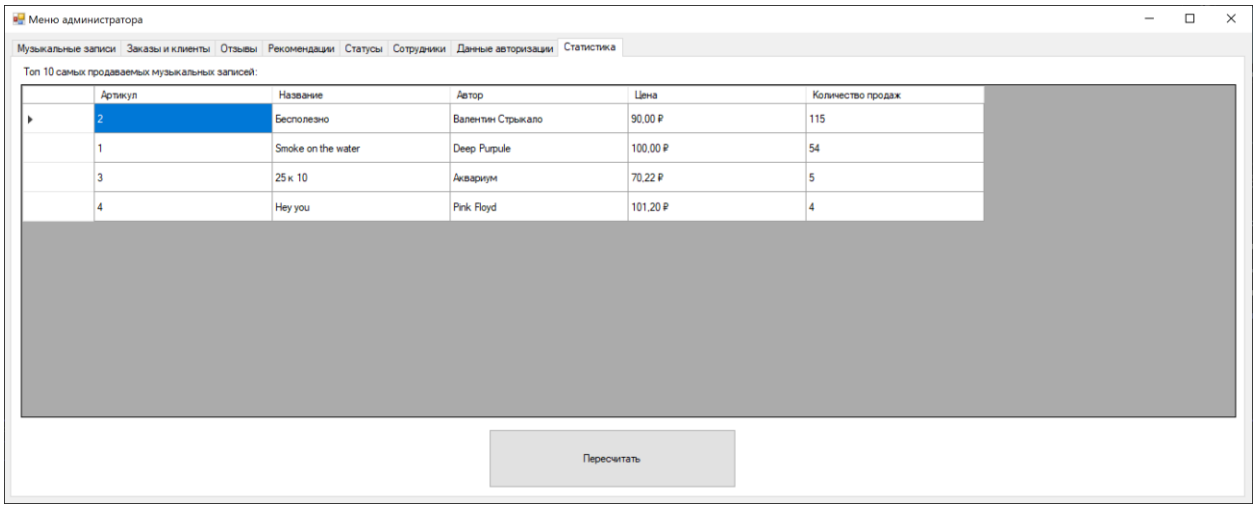


Рисунок 18 Окно статистики для администратора

Однако, кроме этого, администратор может добавлять и изменять данные сотрудников:

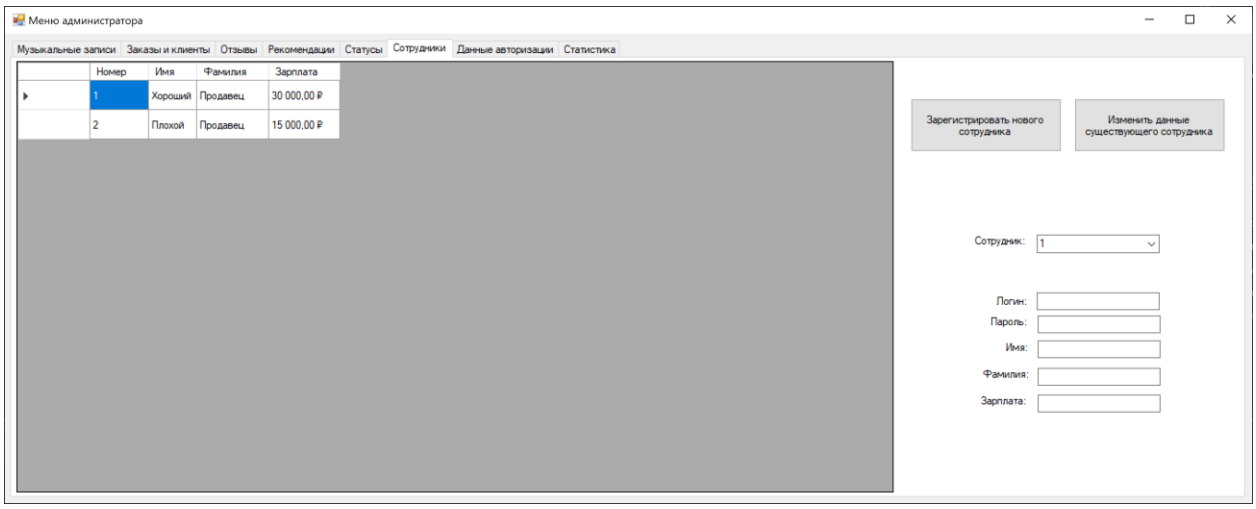


Рисунок 19 Окно сотрудников для администратора

И изменять свои данные авторизации:

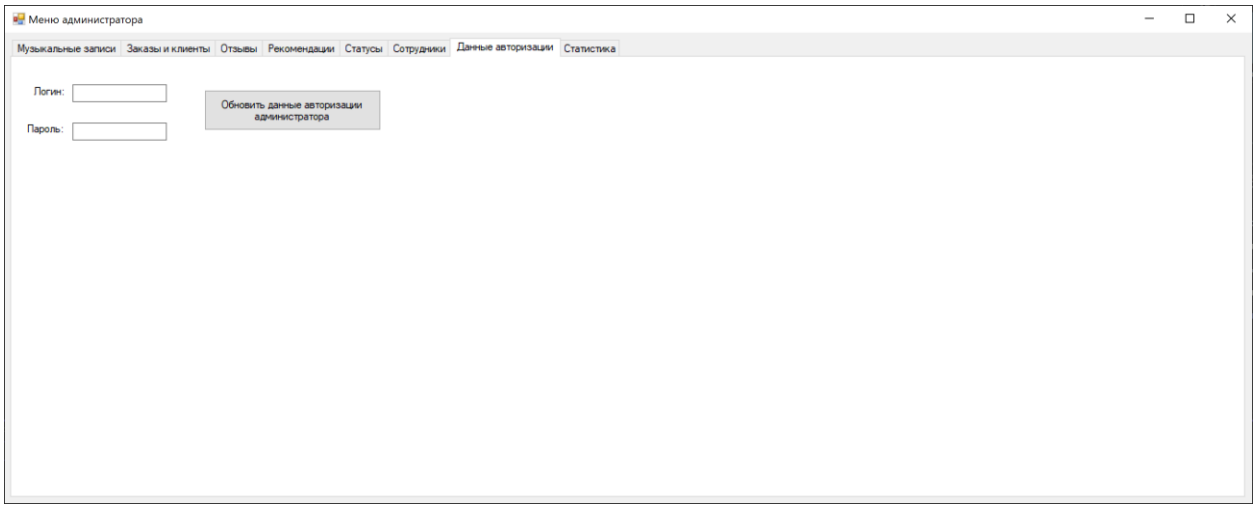


Рисунок 20 Окно изменения данных авторизации для администратора

Заключение

В ходе выполнения курсовой работы была реализована автоматическая система управления салоном музыкальных записей:

- Была создана и задокументирована база данных;

- Были написаны запросы (представления и хранимые процедуры);

- Было написано клиентское приложение на языке C# с использованием .NET Windows Forms;

Благодаря созданному приложению управлять настоящим салоном музыкальных записей станет значительно проще, поскольку БД позаботится о корректном добавлении, хранении, изменении и удалении информации.

Список литературы

1. Руководство по классическим приложениям Windows Forms .NET - https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-5.0
2. Руководство по SQL Server - https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16
3. Справочник по Transact-SQL - https://learn.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver15

Приложение – код создания таблиц

```
USE [Salon]
GO
/****** Object:  Table [dbo].[clients]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[clients](
	[client_id] [int] IDENTITY(1,1) NOT NULL,
	[first_name] [nvarchar](50) NOT NULL,
	[last_name] [nvarchar](50) NOT NULL,
	[email] [nvarchar](50) NOT NULL,
	[address] [nvarchar](50) NOT NULL,
	[login_id] [int] NOT NULL,
	[dob] [date] NOT NULL,
	[phone] [nchar](11) NOT NULL,
 CONSTRAINT [PK_clients] PRIMARY KEY CLUSTERED
(
	[client_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY],
 CONSTRAINT [IX_clients] UNIQUE NONCLUSTERED
(
	[email] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY],
 CONSTRAINT [U_email] UNIQUE NONCLUSTERED
(
	[email] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY],
 CONSTRAINT [UQ__clients__C2C971DA1156DA85] UNIQUE NONCLUSTERED
(
	[login_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[employees]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[employees](
	[employee_id] [int] IDENTITY(1,1) NOT NULL,
	[first_name] [nvarchar](50) NOT NULL,
	[last_name] [nvarchar](50) NOT NULL,
	[salary] [money] NOT NULL,
	[login_id] [int] NOT NULL,
 CONSTRAINT [PK_employees] PRIMARY KEY CLUSTERED
(
	[employee_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY],
 CONSTRAINT [UQ__employee__C2C971DAB16F48A1] UNIQUE NONCLUSTERED
(
```

```
	[login_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[logins]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[logins](
	[login_id] [int] IDENTITY(1,1) NOT NULL,
	[login] [nvarchar](50) NOT NULL,
	[password] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_logins] PRIMARY KEY CLUSTERED
(
	[login_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY],
 CONSTRAINT [IX_logins] UNIQUE NONCLUSTERED
(
	[login] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[orders]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[orders](
	[order_id] [int] IDENTITY(1,1) NOT NULL,
	[client_id] [int] NOT NULL,
	[employee_id] [int] NULL,
	[status_id] [int] NOT NULL,
	[amount] [money] NOT NULL,
	[order_datetime] [datetime] NOT NULL,
 CONSTRAINT [PK_orders] PRIMARY KEY CLUSTERED
(
	[order_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[orders_records]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[orders_records](
	[order_id] [int] NOT NULL,
	[record_id] [int] NOT NULL,
	[count] [int] NOT NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[recommendations]    Script Date: 06.12.2022 17:11:34
******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[recommendations](
	[recommendation_id] [int] IDENTITY(1,1) NOT NULL,
	[employee_id] [int] NOT NULL,
	[client_id] [int] NOT NULL,
	[text] [ntext] NOT NULL,
	[recommendation_datetime] [datetime] NOT NULL,
 CONSTRAINT [PK_recommendations] PRIMARY KEY CLUSTERED
(
	[recommendation_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[recommendations_records]    Script Date: 06.12.2022
17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[recommendations_records](
	[recommendation_id] [int] NOT NULL,
	[record_id] [int] NOT NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[records]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[records](
	[record_id] [int] IDENTITY(1,1) NOT NULL,
	[title] [nvarchar](50) NOT NULL,
	[author] [nvarchar](50) NOT NULL,
	[price] [money] NOT NULL,
	[count] [int] NOT NULL,
 CONSTRAINT [PK_records] PRIMARY KEY CLUSTERED
(
	[record_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[reviews]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[reviews](
	[review_id] [int] IDENTITY(1,1) NOT NULL,
	[record_id] [int] NOT NULL,
	[client_id] [int] NOT NULL,
	[text] [ntext] NOT NULL,
 CONSTRAINT [PK_reviews] PRIMARY KEY CLUSTERED
(
	[review_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

```
/****** Object:  Table [dbo].[statuses]    Script Date: 06.12.2022 17:11:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[statuses](
	[status_id] [int] NOT NULL,
	[text] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_statuses] PRIMARY KEY CLUSTERED
(
	[status_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[clients]  WITH CHECK ADD  CONSTRAINT [FK_clients_logins2] FOREIGN
KEY([login_id])
REFERENCES [dbo].[logins] ([login_id])
GO
ALTER TABLE [dbo].[clients] CHECK CONSTRAINT [FK_clients_logins2]
GO
ALTER TABLE [dbo].[employees]  WITH CHECK ADD  CONSTRAINT [FK_employees_logins1] FOREIGN
KEY([login_id])
REFERENCES [dbo].[logins] ([login_id])
GO
ALTER TABLE [dbo].[employees] CHECK CONSTRAINT [FK_employees_logins1]
GO
ALTER TABLE [dbo].[orders]  WITH CHECK ADD  CONSTRAINT [FK_orders_clients] FOREIGN
KEY([client_id])
REFERENCES [dbo].[clients] ([client_id])
GO
ALTER TABLE [dbo].[orders] CHECK CONSTRAINT [FK_orders_clients]
GO
ALTER TABLE [dbo].[orders]  WITH CHECK ADD  CONSTRAINT [FK_orders_employees] FOREIGN
KEY([employee_id])
REFERENCES [dbo].[employees] ([employee_id])
GO
ALTER TABLE [dbo].[orders] CHECK CONSTRAINT [FK_orders_employees]
GO
ALTER TABLE [dbo].[orders]  WITH CHECK ADD  CONSTRAINT [FK_orders_statuses] FOREIGN
KEY([status_id])
REFERENCES [dbo].[statuses] ([status_id])
GO
ALTER TABLE [dbo].[orders] CHECK CONSTRAINT [FK_orders_statuses]
GO
ALTER TABLE [dbo].[orders_records]  WITH CHECK ADD  CONSTRAINT [FK_orders_records_orders]
FOREIGN KEY([order_id])
REFERENCES [dbo].[orders] ([order_id])
GO
ALTER TABLE [dbo].[orders_records] CHECK CONSTRAINT [FK_orders_records_orders]
GO
ALTER TABLE [dbo].[orders_records]  WITH CHECK ADD  CONSTRAINT
[FK_orders_records_records] FOREIGN KEY([record_id])
REFERENCES [dbo].[records] ([record_id])
GO
ALTER TABLE [dbo].[orders_records] CHECK CONSTRAINT [FK_orders_records_records]
GO
ALTER TABLE [dbo].[recommendations]  WITH CHECK ADD  CONSTRAINT
[FK_recommendations_clients] FOREIGN KEY([client_id])
REFERENCES [dbo].[clients] ([client_id])
GO
ALTER TABLE [dbo].[recommendations] CHECK CONSTRAINT [FK_recommendations_clients]
GO
```

```
ALTER TABLE [dbo].[recommendations]  WITH CHECK ADD  CONSTRAINT
[FK_recommendations_employees] FOREIGN KEY([employee_id])
REFERENCES [dbo].[employees] ([employee_id])
GO
ALTER TABLE [dbo].[recommendations] CHECK CONSTRAINT [FK_recommendations_employees]
GO
ALTER TABLE [dbo].[recommendations_records]  WITH CHECK ADD  CONSTRAINT
[FK_recommendations_records_recommendations] FOREIGN KEY([recommendation_id])
REFERENCES [dbo].[recommendations] ([recommendation_id])
GO
ALTER TABLE [dbo].[recommendations_records] CHECK CONSTRAINT
[FK_recommendations_records_recommendations]
GO
ALTER TABLE [dbo].[recommendations_records]  WITH CHECK ADD  CONSTRAINT
[FK_recommendations_records_records] FOREIGN KEY([record_id])
REFERENCES [dbo].[records] ([record_id])
GO
ALTER TABLE [dbo].[recommendations_records] CHECK CONSTRAINT
[FK_recommendations_records_records]
GO
ALTER TABLE [dbo].[reviews]  WITH CHECK ADD  CONSTRAINT [FK_reviews_clients] FOREIGN
KEY([client_id])
REFERENCES [dbo].[clients] ([client_id])
GO
ALTER TABLE [dbo].[reviews] CHECK CONSTRAINT [FK_reviews_clients]
GO
ALTER TABLE [dbo].[reviews]  WITH CHECK ADD  CONSTRAINT [FK_reviews_records] FOREIGN
KEY([record_id])
REFERENCES [dbo].[records] ([record_id])
GO
ALTER TABLE [dbo].[reviews] CHECK CONSTRAINT [FK_reviews_records]
GO
ALTER TABLE [dbo].[clients]  WITH CHECK ADD  CONSTRAINT [check_phone] CHECK  ((NOT
[phone] like '%[^0-9]%'))
GO
ALTER TABLE [dbo].[clients] CHECK CONSTRAINT [check_phone]
GO
```