МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы

и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных «Кофейня» с применением резервного копирования и восстановления данных»

Выполнил студент Иванова А.А

(Ф.И.О.)

Руководитель проекта ассистент Копыток Д.В

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ассистент Копыток Д.В

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ассистент Копыток Д.В

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2022

**Оглавление**

[Введение 3](#_Toc103186717)

[1 Постановка задачи 5](#_Toc103186718)

[2 Аналитический обзор литературы 6](#_Toc103186719)

[3. Разработка модели базы данных 9](#_Toc103186720)

[4 Разработка необходимых объектов 10](#_Toc103186721)

[5 Описание процедур импорта и экспорта 12](#_Toc103186722)

[6 Тестирование производительности 15](#_Toc103186723)

[7 Описание технологии и ее применения в базе данных 16](#_Toc103186724)

[8 Руководство пользователя 18](#_Toc103186725)

[Заключение 29](#_Toc103186726)

[Cписок источников 30](#_Toc103186727)

[Приложение А 31](#_Toc103186728)

[Приложение Б 33](#_Toc103186729)

[Приложение В 34](#_Toc103186730)

# **Введение**

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных — база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана Oracle 19c, в связи с ее высокой производительностью и надежностью.

Так же было необходимо разработать приложение для демонстрации работы базы данных, взаимодействия с ней. Приложение было написано на языке программирования C# с графическим интерфейсом, выполненным с помощью WPF. Для взаимодействия с сервером базы данных Oracle использовался пакет Oracle Managed Data Access для .NET.

В основной части будут затронуты все аспекты разработки проекта и обоснованы некоторые технические приёмы, к которым приходилось прибегнуть, с целью реализации работы веб-сервера с базой данных.

Любая организация нуждается в своевременном доступе к информации. Ценность информации в современном мире очень высока. Роль распорядителей информации в современном мире чаще всего выполняют базы данных. Базы данных обеспечивают надежное хранение информации, в структурированном виде и своевременный доступ к ней. Практически любая современная организация нуждается в базе данных, удовлетворяющей те или иные потребности по хранению, управлению и администрированию данных.

За последние несколько лет наблюдается тенденция к усложнению структур данных. Простые виды информации, представимой в форме чисел и текстовых строк, не утратив своей значимости, дополняются сегодня многочисленными мультимедийными документами, графическими образами, хронологическими рядами, процедурными, или активными, данными и мириадами прочих сложных информационных форм.

На сегодняшний день на рынке представлено множество технологий доступа к данным и серверов баз данных, каждое, из которых имеет свои отличительные черты. Современные приложения обработки данных ориентированы на работу с большим количеством пользователей, на их удаленность от места расположения основного сервера БД.

Темой данного курсового проекта является разработка программы «Poster», обеспечивающая работу с базой данных «Кафе».

Приложение «Poster» имеет доступ к таблицам в базе данных и предназначено для взаимодействия с источником данных. Взаимодействие подразумевает получение данных, их представление в определенном формате для просмотра пользователем, редактирование в соответствии с реализованными в программе бизнес-алгоритмами и возврат обработанных данных обратно в базу данных.

Автоматизация процессов учета и оформления заказов должна включать в себя разработку системы, обеспечивающую хранение и обработку данных о блюдах, которые предлагает заведение, а также сотрудников, оформляющих заказ.

База данных «Кофейня» существенно упрощает работу сотрудников кофейни, а также предоставляет возможность администратору своевременно вносить необходимые изменения. Данный программный продукт позволяет автоматизировать работу любого заведения ресторанного типа.

Целью данной работы является разработка реляционной базы данных для кофейни: «Кофейня». Эта база данных должна составлялась для обеспечения управления основными задачами кофейни: создавать, изменять и удалять заказы, блюда, которые предлагает кофейня, а также работников кофейни. Так же было необходимо разработать соответствующее приложение, для демонстрации её работы.

# **1 Постановка задачи**

Целью данной курсовой работы является создание программного обеспечения для автоматизации процесса администрирования процессов приема и увольнения сотрудников и создание, и изменение формации о блюдах, предлагаемых кофейней, кроме этого, приложение позволяет администрировать процессы оформления и изменения заказов.

Задача проекта: совершенствование практических навыков в использовании и разработке современных информационных систем, разработка архитектуры приложения, Создание базы данных и интерфейса, выполнение тестирования готового продукта.

Функционально должны быть выполнены следующие задачи:

* разработка приложения для взаимодействия с БД;
* регистрация персонала;
* регистрация новых блюд;
* обновление списка блюд;
* оформление заказов;

Должны быть выполнены следующие требования:

− доступ к данным должен осуществляться только через соответствующие процедуры;

− должен быть проведен импорт данных из XML файлов, экспорт данных в формат XML;

− необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости. Необходимо проанализировать планы запросов к таблице;

− применить технологию базы данных согласно выбранной теме: подробно описать применяемые системные пакеты, утилиты или технологии; показать применение указанной технологии в базе данных.

Десктоп приложение должно быть реализовано с использованием языка С#. Технология для реализации – WPF [2]. База данных приложения должно взаимодействовать с Oracle Database 19с.

Разрабатываемое приложение должно содержать форму авторизации. Следует разработать два уровня доступа: уровень доступа для администратора и уровень доступа для обычных работников заведения. Администратор имеет возможность добавлять сотрудников, блюда, а также изменять информацию о них.

Результатом данной курсовой работы является проект, который будет осуществлять взаимодействие с разработанной базой данных «Кофейня».

В процессе выполнения работы были поставлены следующие задачи:

1. Провести аналитический обзор литературы, а также поиск и сравнение аналогов.
2. Спроектировать и разработать программное средство и базу данных.
3. Провести тестирование разработанного программного средства и написать руководство пользователя разработанного программного средства.
4. Провести тестирование производительности программного обеспечения

# **2 Аналитический обзор литературы**

Объектом разработки является приложение для менеджмента процессов кофейни. Кофейня — гастрономическое заведение, место для встреч и общения, где подают напитки и десертные блюда. Заведения подобного плана чаще всего являются местом для деловых встреч, работы, отдыха за счет своей непринужденной обстановки. У кофеен достаточно много предназначений, одно из них — дать возможность немного отдохнуть, выпивая качественный кофе, занятым и спешащим горожанам. Бизнес постепенно выходит в виртуальную сферу. Количество кофеен, ведущих свою деятельность в интернет-пространстве, за последние 5 лет увеличилось на 40%. Основная задача курсовой работы – создание базы данных «Кофейня», а также максимально удобное приложение, которое будет осуществлять работу с этой базой данных.

При разработке своего приложения я обратила внимания на самые успешные примеры популярнейших приложений. Рассмотрим их ниже.

Сегодня Iiko – крупнейший в мире сервис, позволяющий администрирование работы кофеен и ресторанов. Свое признание он получил преимущественно в Европе, однако, в Беларуси им пользуется около десятка заведений. Интерфейс этого сервиса можно увидеть на рисунке 2.1

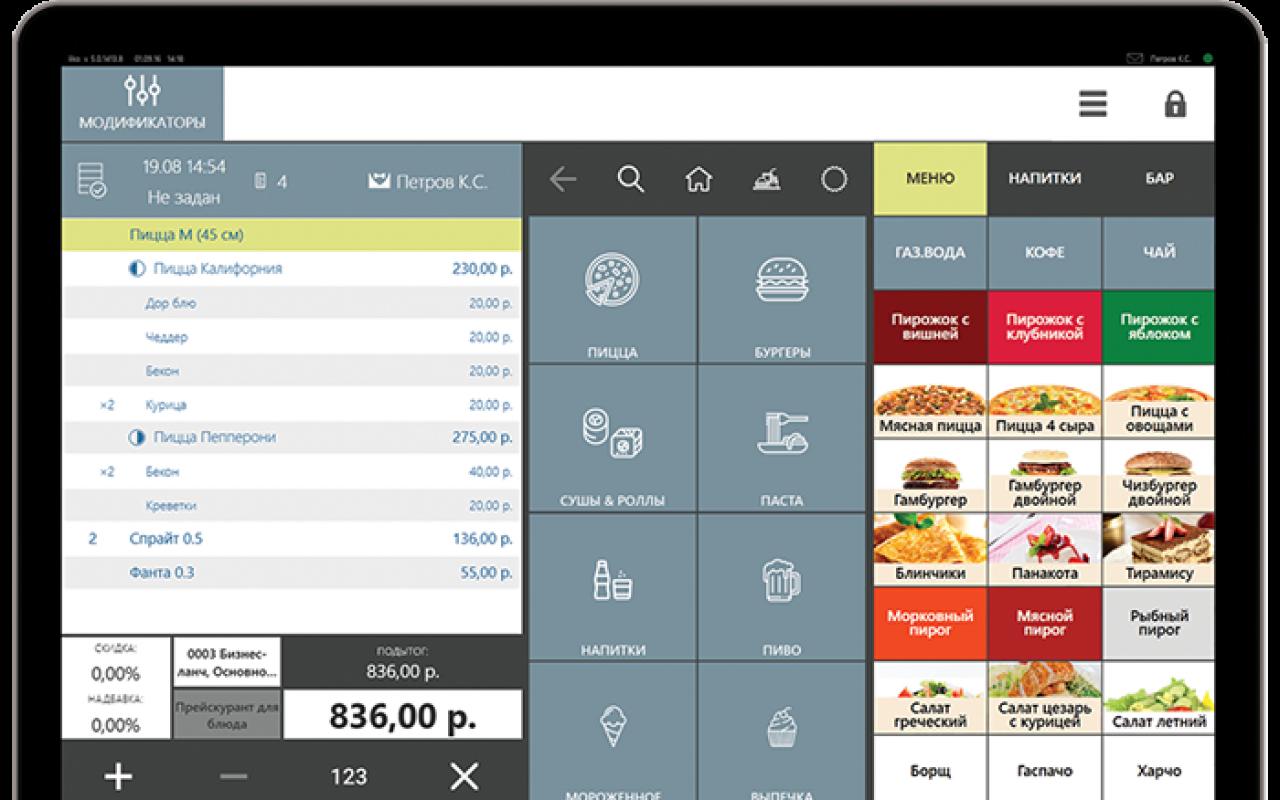


Рисунок 2.1 – Главная страница приложения «Iiko»

Следующее, но не менее популярное приложение – Rkeeper. Данное приложение создавалось изначально для заведений, не предполагающих большой поток людей, однако свое признание оно получило в сфере ресторанного бизнеса, нежели кофеен или кафе из-за переизбыточности функций. Скриншот главной страницы RKeeper предоставлен на рисунке 2.2.

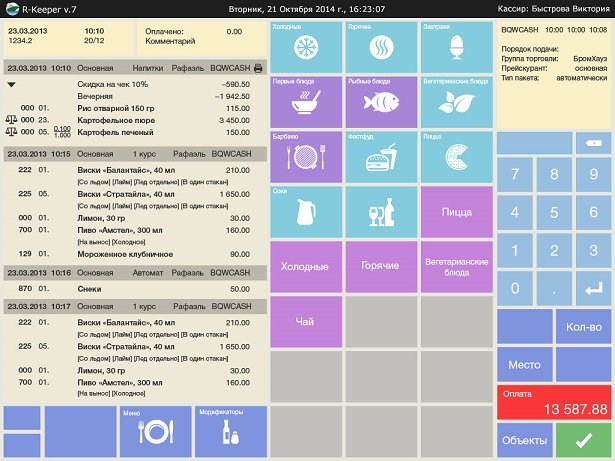


Рисунок 2.2 – главная страница приложения «RKeeper»

Приложение PosJam, данное приложение получило свое призвание в Соединенных Штатах Америки. Данное приложение является как десктопным приложением, так и мобильным. Ключевым преимуществом данного приложения является облачное хранение данных на google cloud, что позволяет контролировать работу заведений во всех уголках мира и не заботится о собственных стационарных серверах. Главный экран данного приложения изображен на рисунке 2.3

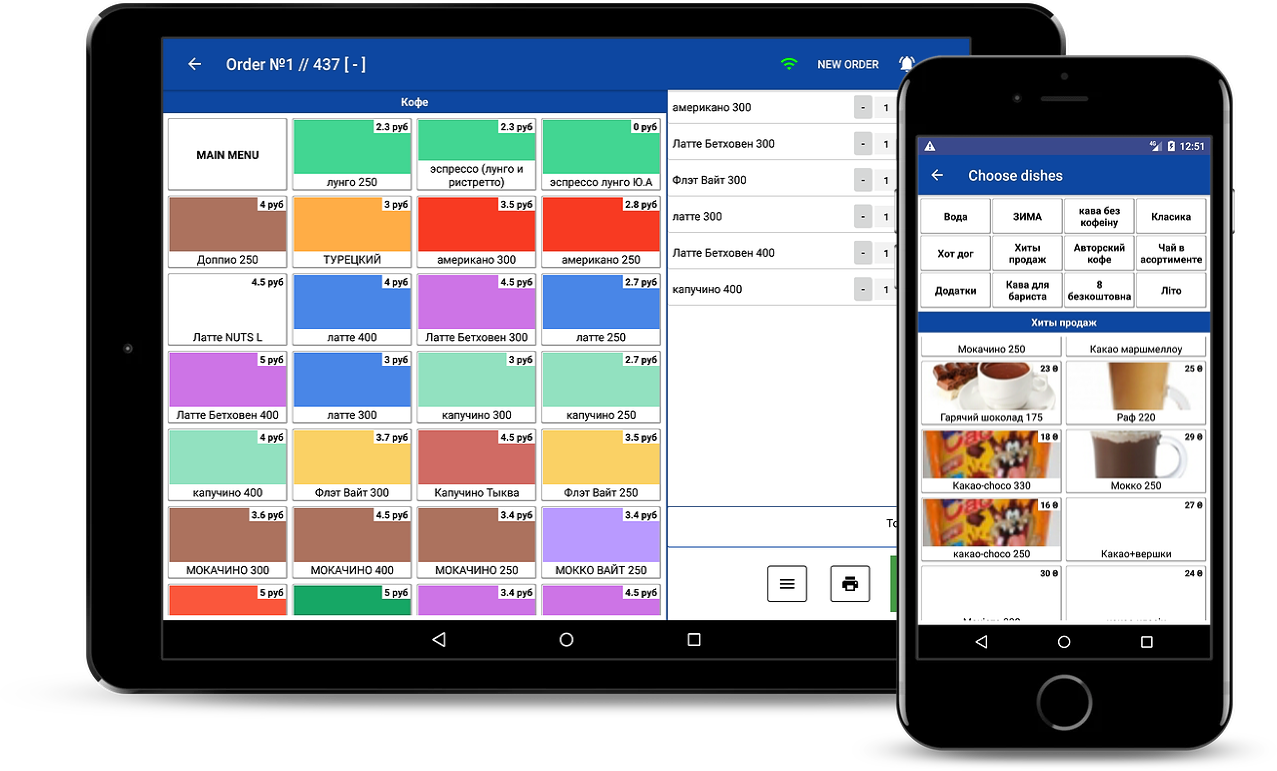


Рисунок 2.3 – Домашняя страница приложения PosJam

Разобрав три, приведенных выше, сервиса для администрирования заведений ресторанного типа, можно сформировать общий функционал необходимый для работы приложения.

Прежде всего, естественно, база данных должна корректно работать с большим количеством данных, при этом не быть перегружена. Обязательно должна быть функция выставления скидки в заказе, и отображения всего списка блюд в заказе.

Также стоит ввести возможность просмотра простейшей статистики заказов за определенные промежутки времени для администратора.

# **Разработка модели базы данных**

Первым этапом курсовой работы будет создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуры нашей базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей [6].

Диаграмма базы данных, спроектированной в ходе разработки приложения представлена на рисунке 3.1.

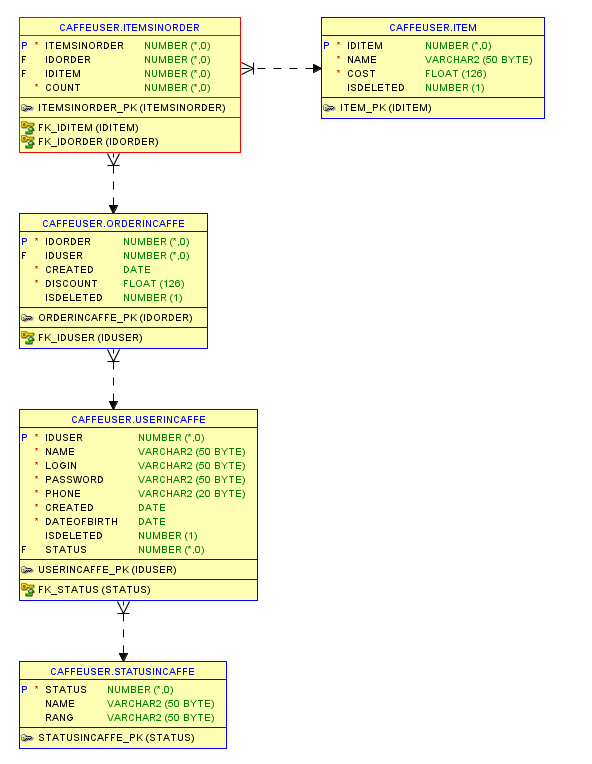
****

Рисунок 3.1 – Диаграмма базы данных

База данных состоит из следующих таблиц:

* Item;
* ItemsInOrder;
* OrderInCaffe;
* UserInCaffe;
* StatusInCaffe;

# **4 Разработка необходимых объектов**

В ходе проектирования базы данных было создано 5 таблиц, связанных между собой внешними ключами. Скрипты создания таблиц хранятся в приложении А, скрипты создания пользователей хранятся в приложении Б.

Таблица «Item» хранит список блюд, которые кофейня предлагает посетителям. Поля таблицы представлены в таблице 4.1.

Таблица 4.1 – Столбцы таблицы Item

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| IDITEM | number | идентификатор товара, первичный ключ |
| NAME | varchar2(50) | название товара |
| COST | float | стоимость товара |
| ISDELETED | number | индикатор, означающий удален ли товар |

Таблица «ItemsInOrder» содержит список всех товаров, которые были заказаны посетителями, а также их количество. Поля таблицы представлены в таблице 4.2.

Таблица 4.2 – Столбцы таблицы ItemsInOrder

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| ITEMSINORDER | number | идентификатор объекта, первичный ключ |
| IDORDER | number | идентификатор заказа, которому принадлежит данный заказ, внешний ключ к таблице «OrderInCaffe» |
| IDITEM | number | идентификатор блюда, которое заказал посетитель, внешний ключ к таблице «Item» |
| COUNT | number | количество блюд в заказе |

Таблица «OrderInCaffe» содержит список заказов. Важно понимать, что в данной таблице нет списка блюд, которые заказал гость, они находятся в таблице «ItemsInOrder» Поля таблицы представлены в таблице 4.3.

Таблица 4.3 – Столбцы таблицы OrderInCaffe

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| IDORDER | number | идентификатор заказа, первичный ключ |
| IDUSER | nvarchar2(50) | идентификатор пользователя, который создал/изменил заказ, внешний ключ к таблице «UsersInCaffe» |
| CREATED | date | дата создания или изменения заказа |
| DISCOUNT | float | размер скидки на заказ |
| ISDELETE | number | индикатор, означающий удален ли заказ |

Таблица «StatusInCaffe» содержит в себе список статусов, а также уровень привилегий статуса. Такая таблица позволит динамически расширять количество статусов в базе данных, что позволяет модифицировать полномочия пользователей приложения. Поля таблицы представлены в таблице 4.4.

Таблица 4.4 – Столбцы таблицы StatusInCaffe

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| IDSTATUS | number | идентификатор статуса, первичный ключ |
| NAME | nvarchar2(50) | название статуса |
| RANG | nvarchar2(50) | уровень доступа |

Таблица «UserInCaffe» содержит данные о сотрудниках, которые имеют доступ к системе. Поля таблицы представлены в таблице 4.5.

Таблица 4.5 – Столбцы таблицы UserInCaffe

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| IDUSER | number | идентификатор аккаунта, первичный ключ |
| NAME | varchar2(50) | имя сотрудника |
| PHONE | varchar2(50) | телефон сотрудника |
| CREATED | date | дата создания учетной записи |
| DATEOFBIRTH | date | дата рождения сотрудника |
| ISDELETED | number | индикатор, означающий уволен ли сотрудник |
| LOGIN | nvarchar2(50) | логин сотрудника |
| PASSWORD | nvarchar2(50) | пароль сотрудника |
| IDSTATUS | number | идентификатор статуса пользователя, который является сотрудником кофейни, внешний ключ к таблице «StatusInCaffe» |

Следует отметить, что поля «ISDELETED» являются обязательными и неизбежными, исходя из налогового кодекса Республики Беларусь, а также некоторых правил бухгалтерии, из баз данных нельзя удалять блюда, заказы, а также уволенных сотрудников по причине того, что данные таблицы используются в ежеквартальных отчетах, а значит влияют на количество выручки или качество отчетности по оформленным заказам за квартал.

Помимо таблиц в базе данных были созданы объекты: хранимые процедуры для выполнения основных действий с данными в базе данных, функции, последовательности и пользователи.

Все процедуры, связанные с таблицами и пользователями хранятся в отдельных пакетах процедур. Все select запросы хранятся в функциях, которые создают pipeline. Скрипты функций и процедур хранятся в приложении В.

В каждой из процедур, для избежания исключительных ситуаций, происходит обработка исключений с выводом сообщения об ошибке.

Для разграничения прав доступа было создана два пользователя. Один является администратором базы данных. От его имени создавались объекты базы данных. Администратор имеет возможность добавлять сотрудников, сеансы и фильмы. Второй пользователь имеет ограниченные права на выполнение процедур, которые ему выдал администратор.

Также была создана последовательность для нумерации заказов, блюд и пользователей, которая каждому новому объекту ставит идентификатор на единицу больше предыдущего[5].

# **5 Описание процедур импорта и экспорта**

Импорт и экспорт данных — это автоматический или полуавтоматический ввод и вывод наборов данных между различными программными приложениями. Он включает «перевод» из формата, используемого в одном приложении, в формат, используемый другим, когда такой перевод выполняется автоматически через машинные процессы, такие как [транскодирование](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1), преобразование данных и другие. Истинный экспорт данных часто содержит данные в необработанных форматах, которые в противном случае не могут быть прочитаны конечными пользователями без пользовательского интерфейса, который был разработан для их отображения. В моей курсовой работе импорт и экспорт данных был произведен как с помощью JSON, так и с помощью XML форматов. Для импорта более 100000 строк было решено использовать именно JSON формат. Файлы были созданы с помощью веб-генератора JSON файлов. Пример такого файла представлен на рисунке 5.1

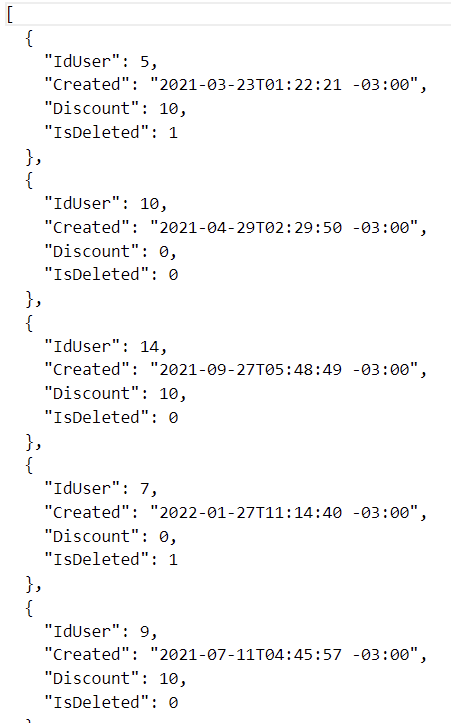


Рисунок 5.1 – Пример JSON файла, сгенерированный веб-генератором

Далее, с помощью python был написан скрипт (листинг 5.1), с помощью которого все строки были импортированы в базу данных

|  |
| --- |
| import cx\_Oracle  import json  connection = cx\_Oracle.connect(      user="CaffeUser",      password="secret",      dsn="localhost/ORCL")  print("Successfully connected to Oracle Database")  cursor = connection.cursor()  with open('ItemsInOrder.json', 'r', encoding='utf-8') as j:      data = json.load(j)  i = 0  for item in data:      cursor.execute("""               begin                  insert into CaffeUser.ItemsInOrder(IdOrder, IdItem, Count)                  values ({0}, {1}, {2});              end;          """.format(item['IdOrder'], item['IdItem'], item['Count']))      i += 1      print('Inserted {0} row'.format(i))      if i % 100 == 0:          connection.commit()  connection.commit() |

Листинг 5.1 – Скрипт импорта данных из json файлов в базу данных, написанный на языке программирования python

XML — расширяемый [язык разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8). Рекомендован [Консорциумом Всемирной паутины](https://ru.wikipedia.org/wiki/W3C) (W3C). Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным [синтаксисом](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), удобный для создания обработки документов как программами, так и человеком, с акцентом на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. Это формат данных, используемый для обмена данными в форме, которая может быть легко использована и распространена. Часто возникает необходимость импортировать и экспортировать XML-файлы в Oracle [4].

В данном курсовом проекте используются пакеты DBMS\_XMLPARSER и DBMS\_XSLPROCESSOR для импорта XML, и DBMS\_XMLDOM для экспорта данных в XML формат. Для работы с файлами так же использовался пакет DBMS\_LOB [3]. В данном курсовом проекте функции экспорта, импорта используются для таблиц:

* OrderInCaffe;
* ItemsInOrder

Пример создания процедуры экспорта можно посмотреть в приложении Г. Скриншот XML файла изображен на рисунке 5.2



Рисунок 5.2 – Скриншот сгенерированного xml файла

Для импорта данных использовались пакеты DBMS\_XMLPARSER и DBMS\_XSLPROCESSOR.

# **6 Тестирование производительности**

Для тестирования производительности разработанного приложения были проведены тесты использование памяти и использование ЦП.

Тестирование использование памяти проводится с целью обнаружить проблему утечки памяти [1].

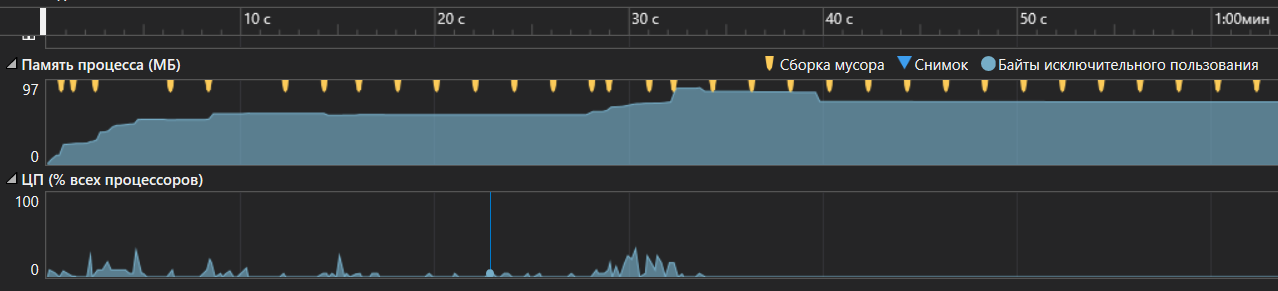
****

Рисунок 6.1 – Тестирование памяти

Как показано на рисунке 6.1 тестирование приложения на проблему утечку памяти пройдено успешно. Приложения в процессе своей работы использует 97 МБ памяти.

Тестирование использование ЦП проводится с целью нахождения степени загруженности центрального процессора при работе программы.

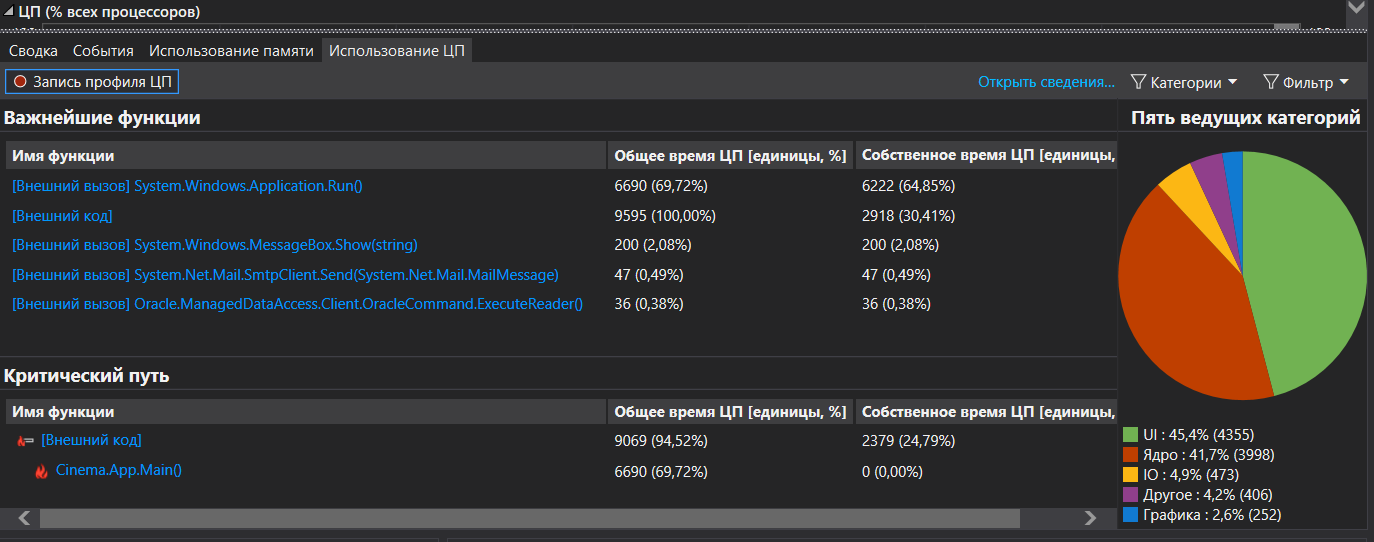


Рисунок 6.2 – Тестирование ЦП

Как показано на рисунке 6.2 тестирование приложения на загруженность процессора основная нагрузка на приложение (69%) при запуске приложения. Стоит заметить, что нагрузка на процессор не постоянная, а в среднем 0.5-0.9 секунды. С учетом того, что все страницы генерируются из базы данных, данный результат можно считать хорошим.

# **7 Описание технологии и ее применения в базе данных**

В процессе разработки проекта была разработана технология: резервное копирование и восстановление данных.

Традиционный пользовательский метод резервного копирования состоит в применении команд операционной системы для копирования необходимых файлов в другое место и/или на ленточное устройство. Резервное копирование – процесс создания копии данных на носителе, предназначенном для восстановления данных в оригинальном или новом месте их расположения в случае их повреждения или разрушения. В случае применения утилиты RMAN резервное копирование файлов базы данных Oracle выполняться внутри базы данных посредством самого сервера баз данных.

Диспетчер восстановления Oracle или утилита RMAN, командная строка и инструмент на базе Oracle Enterprise Manager – метод резервного копирования и восстановления БД, рекомендуемый компанией Oracle. Утилита RMAN создана для работы непосредственно с сервером. Утилита позволяет определять поврежденные блоки данных во время резервного копирования и восстановления БД. Утилита RMAN оптимизирует производительность и потребление памяти во время создания бэкапа путем уплотнения файлов и сжатия набора резервных копий.

RMAN умеет делать резервные копии и копии образов файлов данных, управляющих файлов, архивных журналов повторного выполнения, файлов SPFILE. Утилита RMAN восстанавливает файлы данных из резервных копий (backup) и применяет необходимые архивные журналы повторного выполнения для приведения базы данных в актуальное состояние.

Утилита RMAN упрощает процедуры резервного копирования, позволяя использовать мощные и в то же время легкие в написании сценарии резервного копирования и восстановления. Вдобавок RMAN обладает функциям наподобие обнаружения повреждений внутри блоков данных и способностью выполнять резервное копирование только изменившихся блоков в базе данных. Сценарии RMAN можно сохранять в базе данных и использовать прямо оттуда, что избавляет от необходимости писать сценарии уровня операционной системы. RMAN автоматически обеспечивает создание резервных копий всех файлов базы данных, что исключает вероятность допущения человеческой ошибки, которая присутствует в методах резервного копирования, основанных на применении команд операционной системы.

Существует два вида резервного копирования: «холодное» и «горячее».

Из «холодной» резервной копии можно восстановить только то состояние базы данных, которое было в момент останова; транзакции, сделанные после рестарта базы, в «холодную» резервную копию не попадут.

«Горячее» резервное копирование подразумевает копирование файлов базы данных без остановки базы. Поскольку запись данных в базу и чтение файлов БД никак не синхронизированы, программа резервного копирования может прочитать некорректную страницу, в которой половина будет от старой версии страницы, а другая половина – от новой.

В курсовом проекте используется «холодное» резервное копирование. Рассмотрим последовательность команд для создания резервных копий базы данных с помощью утилиты RMAN:

* $rman – запуск утилиты RMAN;
* connect target "admin/Password1@//localhost:1521/RestorePDB.be.by as sysdba" – подключение к целевой базе данных;
* shutdown immediate – немедленная остановка базы данных;
* startup mount – запуск базы данных в режиме MOUNT;
* backup database tag "full\_database\_datafiles" – создание резервной копии файлов данных;
* backup current controlfile tag "full\_database\_controlfile" – создание резервной копии управляющего файла.

Восстановление данных:

* shutdown immediate;
* startup nomount;
* restore database – восстановление файлов базы данных из резервной копии;
* recover database – восстановление данных;
* startup mount;
* alter pluggable database open – запуск базы данных.

# **8 Руководство пользователя**

При запуске приложения появляется окно авторизации (рисунок 8.1).

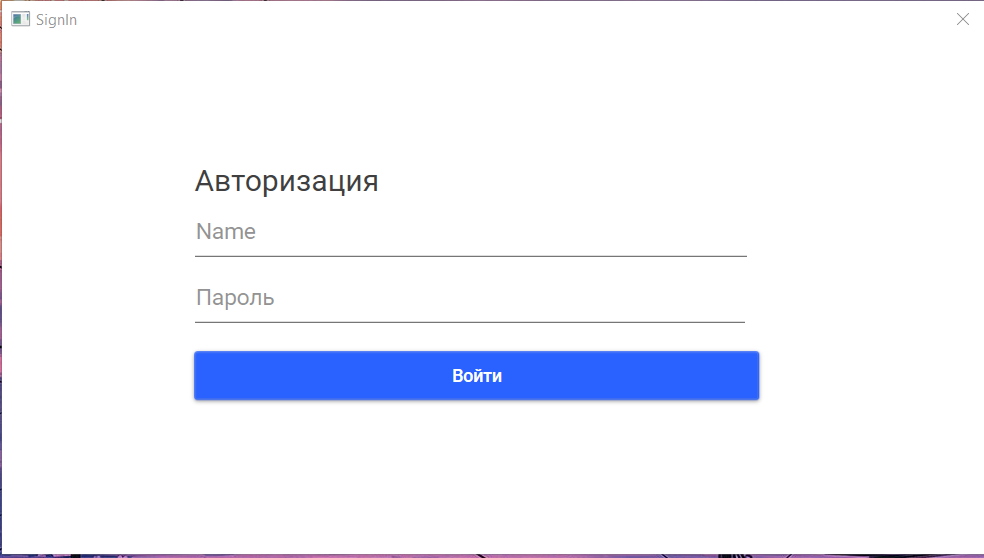


Рисунок 8.1 – Окно авторизации

Можно войти от имени администратора или от имени сотрудника кофейни. При входе от имени администратора открывается окно администратора, где явно указывается ФИО сотрудника (рисунок 8.2).

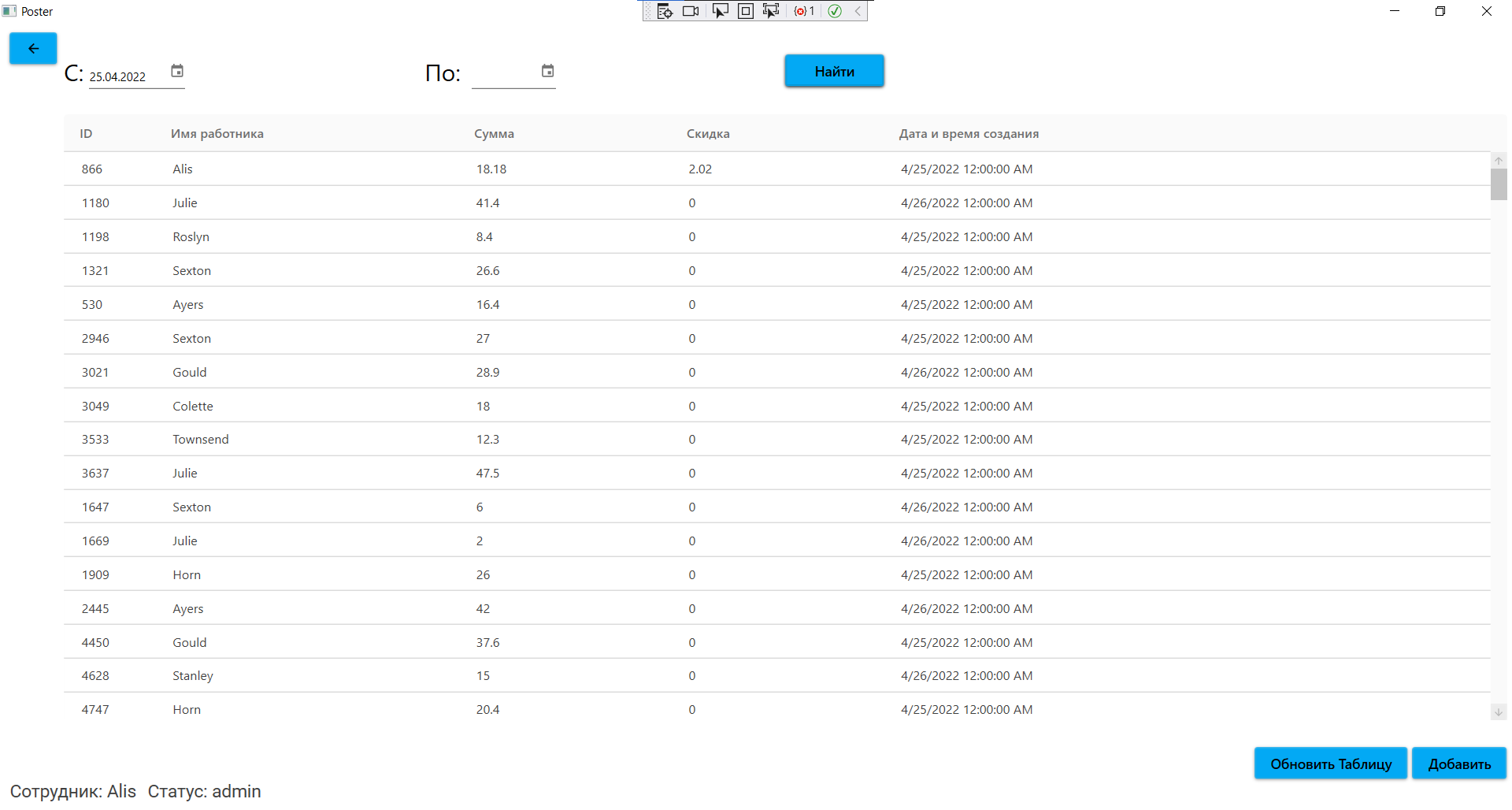


Рисунок 8.2 – Окно администратора

В данном окне, при нажатии на голубую кнопку со стрелкой сверху, есть три вкладки: заказы, позиции, сотрудники, статистика и выход из приложения (рисунок 8.3). Эти вкладки обеспечивают работу со всеми таблицами базы данных. Кнопка выход перенаправит на главную страницу с авторизацией, сделана эта кнопка чтобы сотрудники могли переавторизовываться без необходимости перезапуска приложения.

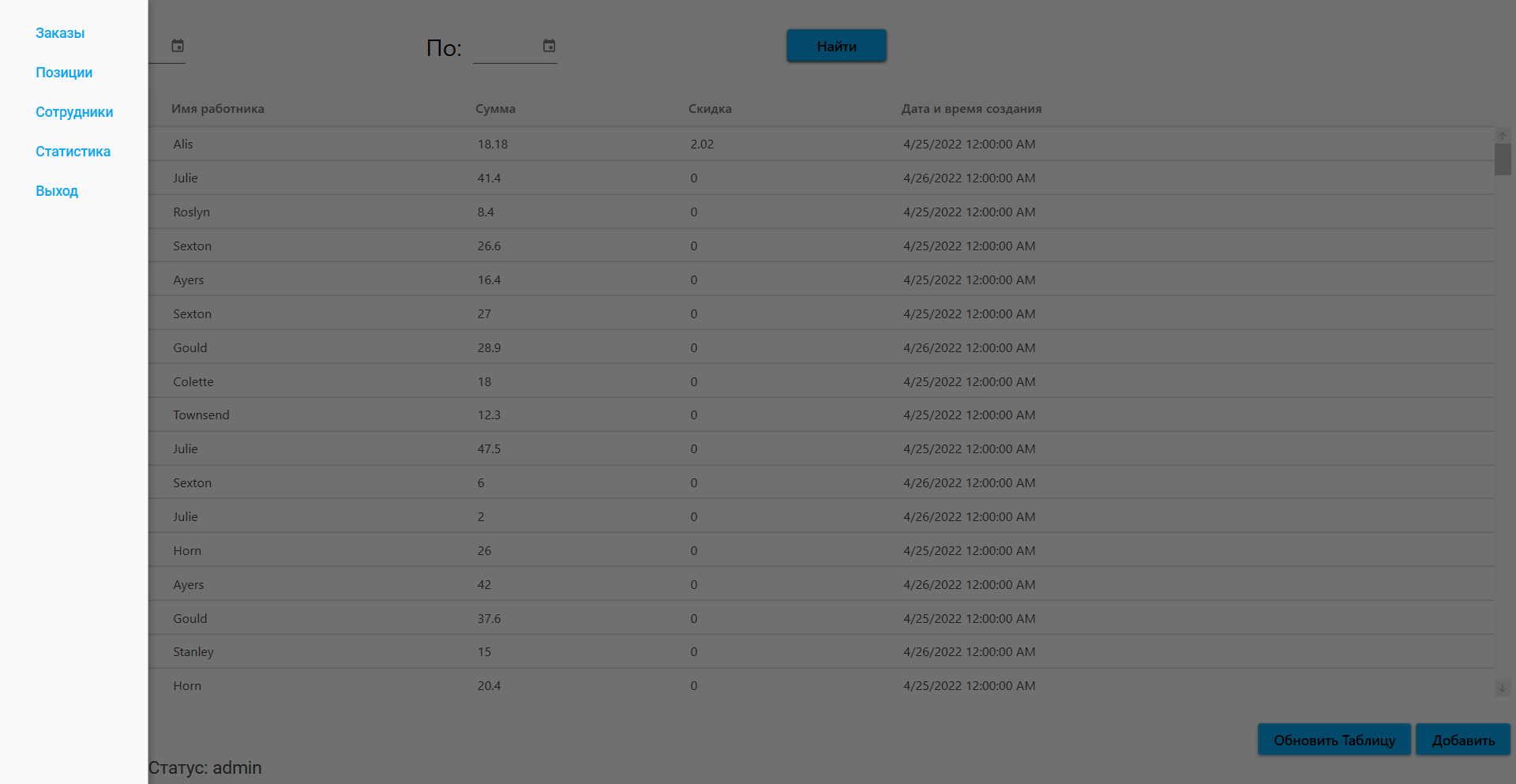


Рисунок 8.3 – окно вкладок приложения

На вкладке сотрудники содержится список всех сотрудников кофейни, а также самая важная информация о них: идентификационный номер, имя, телефон, дата рождения и дата создания учетной записи. Эта информация является одной из самых востребованных, поэтому чтобы у администратора не было необходимости открывать личную карту сотрудника, они вынесены в таблицу. Также здесь есть возможность добавления или изменения сотрудника (рисунок 8.4).

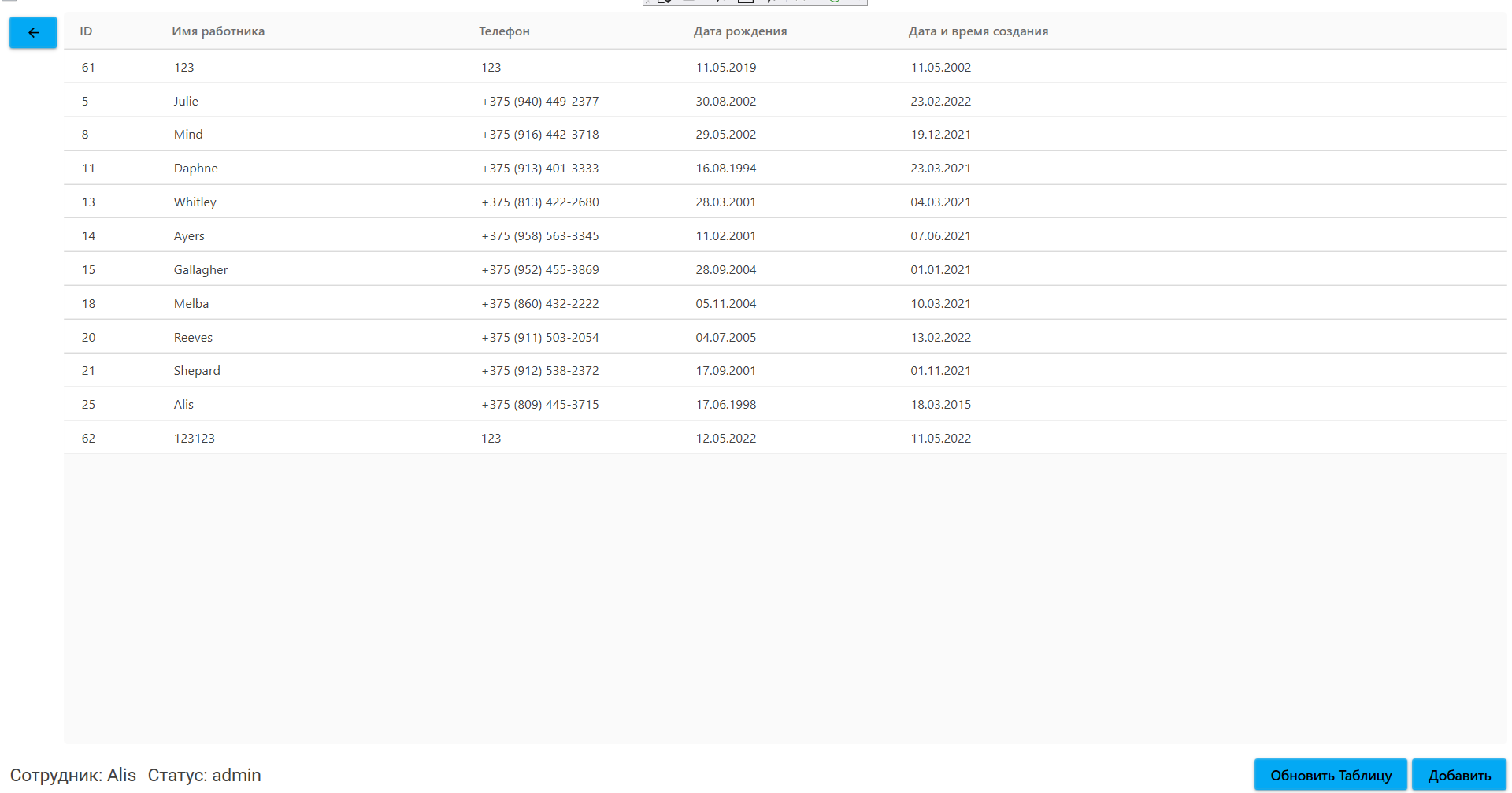


Рисунок 8.4 – Окно списка сотрудников

На этом окне есть кнопки «Добавить» и «Обновить таблицу». Нажав на кнопку «Обновить таблицу», таблица с сотрудниками будет обновлена. Нажав на кнопку «Добавить», появляется окно добавления сотрудника (рисунок 8.5). В данном окне имеется возможность добавления полной информации о сотруднике, которая была видна в таблице с сотрудниками, а также статус, логин и пароль.

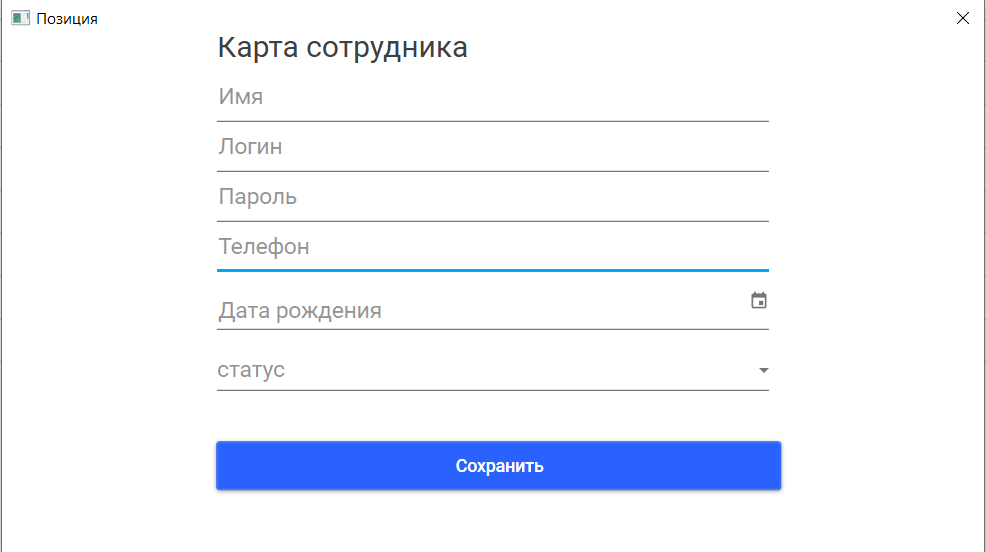


Рисунок 8.5 – Окно добавления сотрудника

В данном окне необходимо заполнить всю информацию и нажать кнопку «Сохранить» (рисунок 8.6). При необходимости выставления статуса высоким уровнем доступа, необходимо выбрать статус «admin»

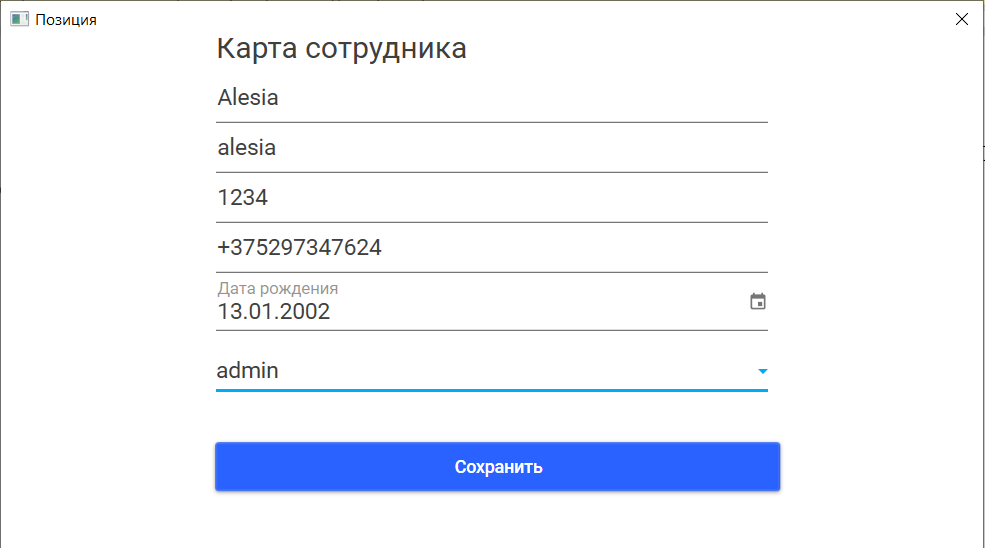


Рисунок 8.6 – Регистрация сотрудника с высоким уровнем доступа

При регистрации пользователей с низким уровнем доступа необходимо выставить статус «user». Сотрудник будет успешно добавлен в базу данных и отобразится в списке на вкладке «Сотрудники» после нажатия кнопки «Обновить Таблицу» или если зайти еще раз на страницу (рисунок 8.7).

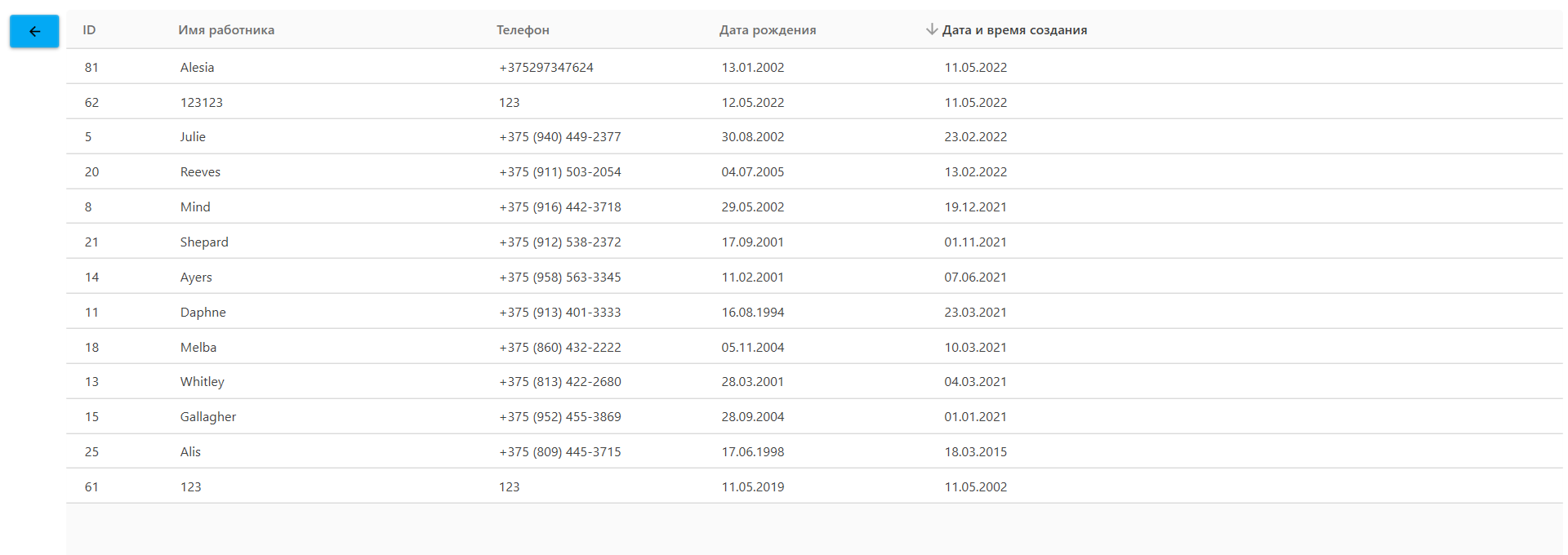


Рисунок 7.7 – Отображение добавленного сотрудника

После добавления сотрудника его данные можно редактировать, дважды щелкнув по соответствующей записи в таблице (рисунок 8.8). Важно отметить, что в данном окне кроме изменения основных данных, можно изменять еще и логин, пароль и статус. В данном окне можно также удалить сотрудника, если его уволили, для этого необходимо нажать на кнопку «Удалить».

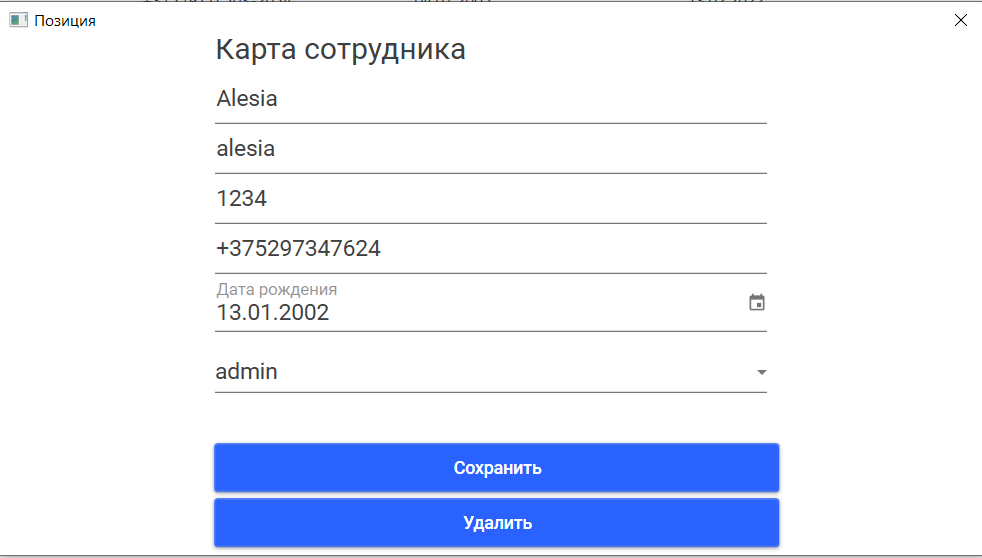


Рисунок 8.8 – Изменение данных сотрудника

После внесения поправок можно сохранить изменения нажав на кнопку «Сохранить».

На вкладке «Позиции» отображается список блюд, которые предоставляет кофейня. В данной вкладке есть возможность сортировке блюд по разным критериям, с помощью двойного нажатия на заголовок таблицы, а также добавление продукта с помощью кнопку «Добавить», его изменение, с помощью двойного нажатия п позиции в таблице, а также обновление таблицы с помощью кнопки «Обновить Таблицу» (рисунок 8.9).

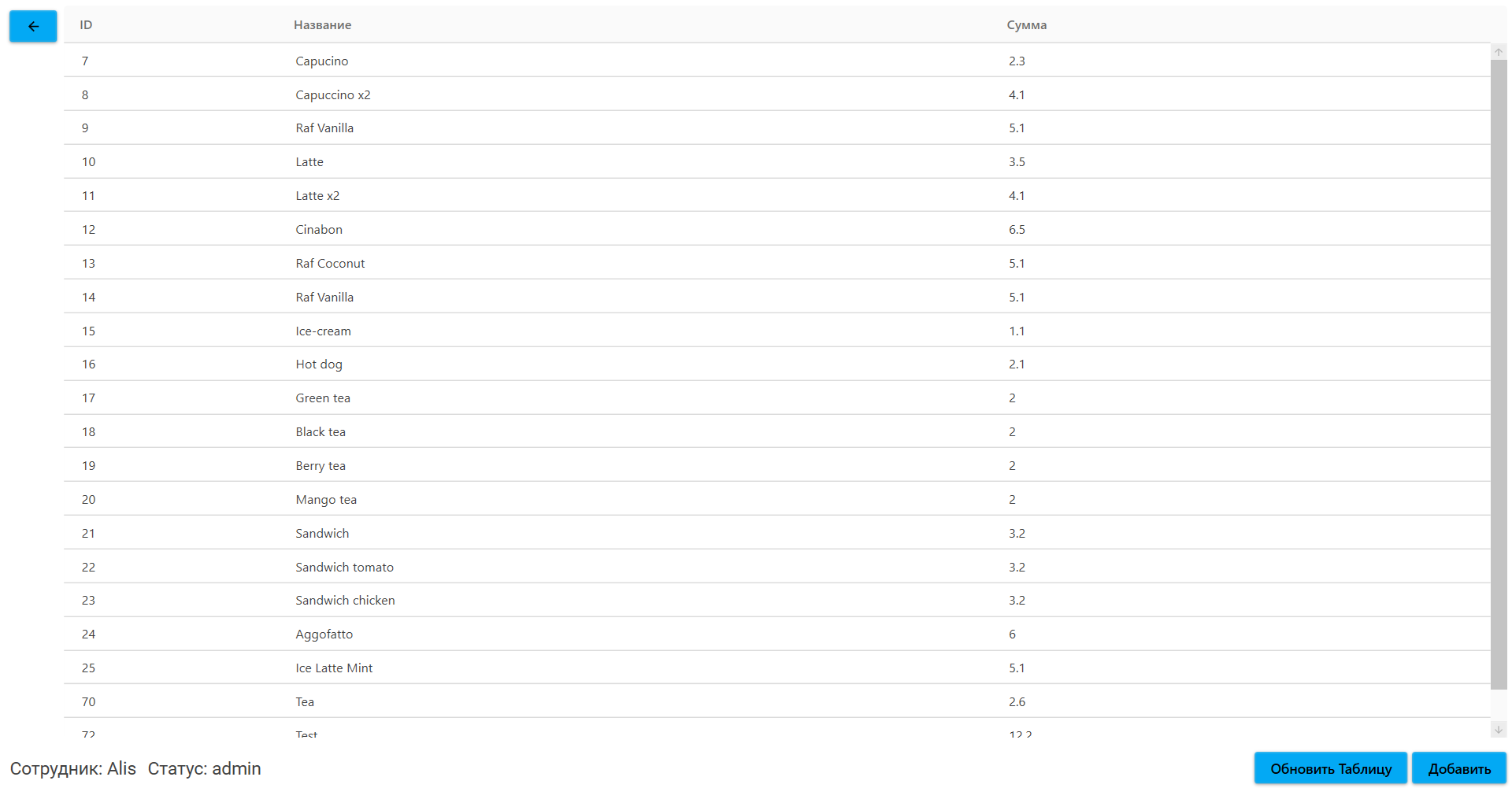


Рисунок 8.9 – Список позиций, сортированный по идентификатору

Позиции, как и сотрудников можно добавлять, изменять и удалять. Данные в таблицах будут изменены при нажатии кнопки «Обновить Таблицу».

Окно добавления позиции представлено на рисунке 8.10. Следует отметить что поле, принимающее сумму, принимает только числовой формат

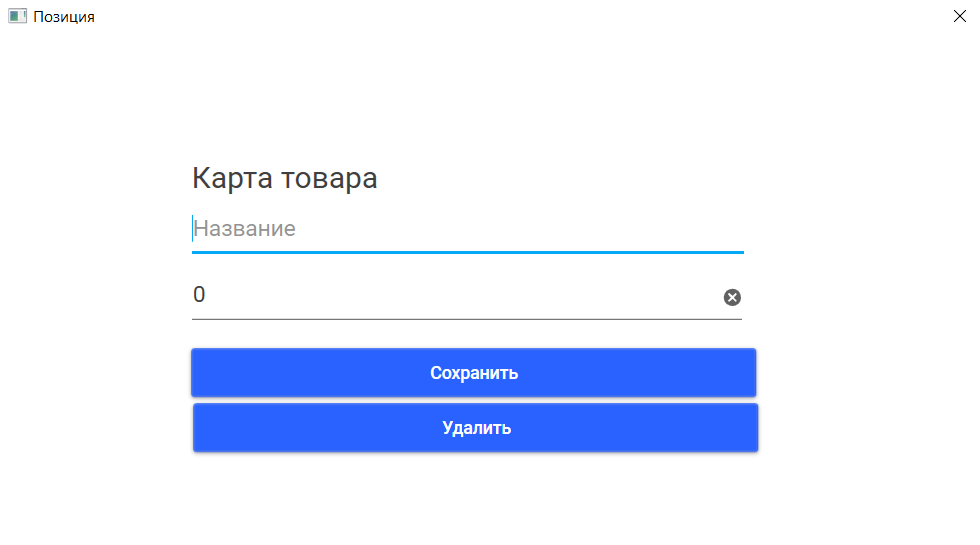


Рисунок 8.10 – Добавление позиции

Окно изменения информации о позиции идентично окну добавления позиции. Интерфейс данного окна представлен на рисунке 8.11.

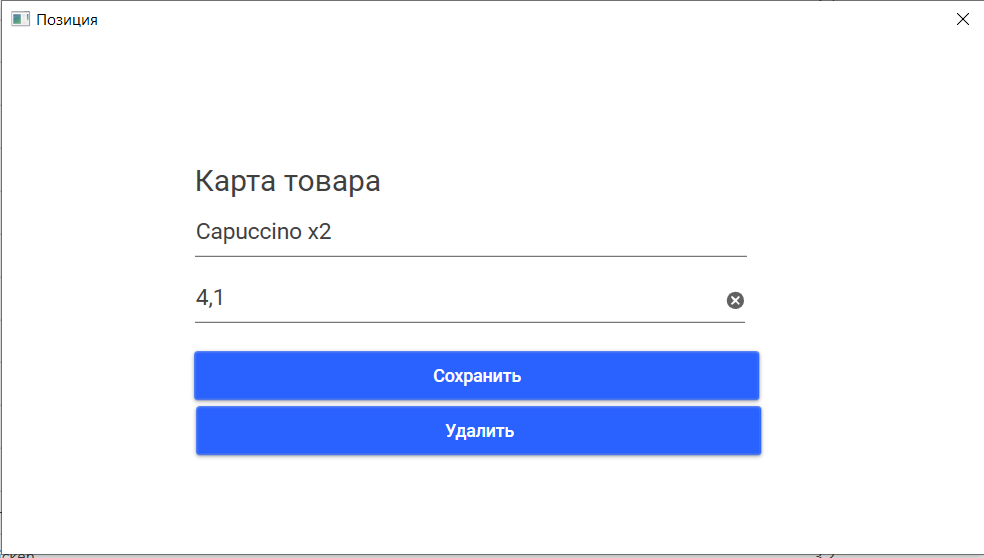


Рисунок 8.11 – Изменение позиции

Удаление сеанса осуществляется в этом же окне: после нажатия на кнопку «Удалить» из всего списка сеансов будет удалена позиция

На вкладке «Заказы» находится список заказов, которые можно отсортировать по датам создания (рисунок 8.12). По умолчанию в таблице показываются заказы за текущий день, при необходимости, можно осуществить поиск заказов в пределах разных дат.

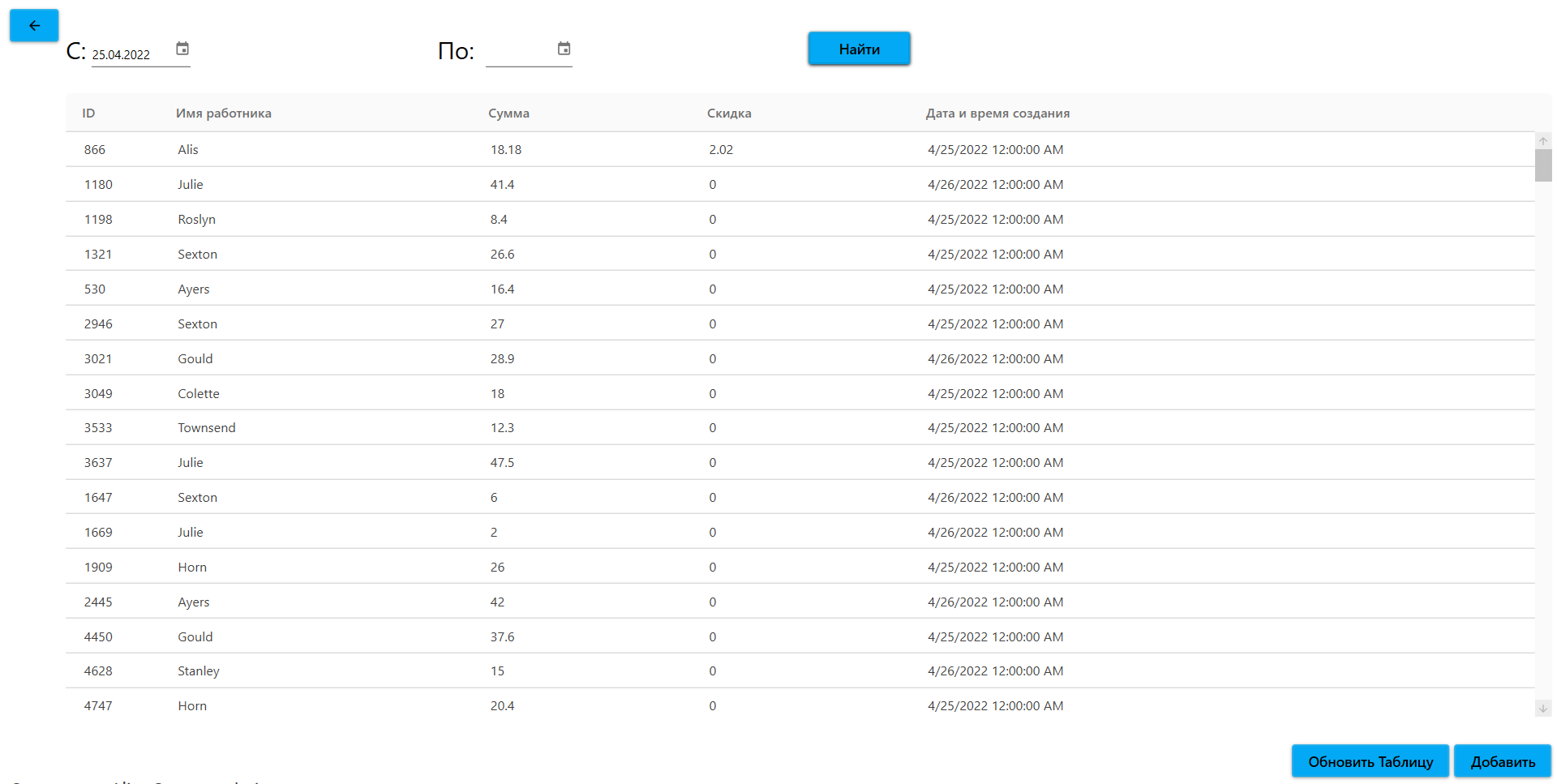


Рисунок 8.12 – Окно списка заказов

Также можно добавить новый заказ, щелкнув по кнопке «Добавить Заказ», в окне редакции заказа вы увидете полный список блюд, которые предлагает кофейня в виде белых кнопок с синией окантовкой (рисунок 8.13).

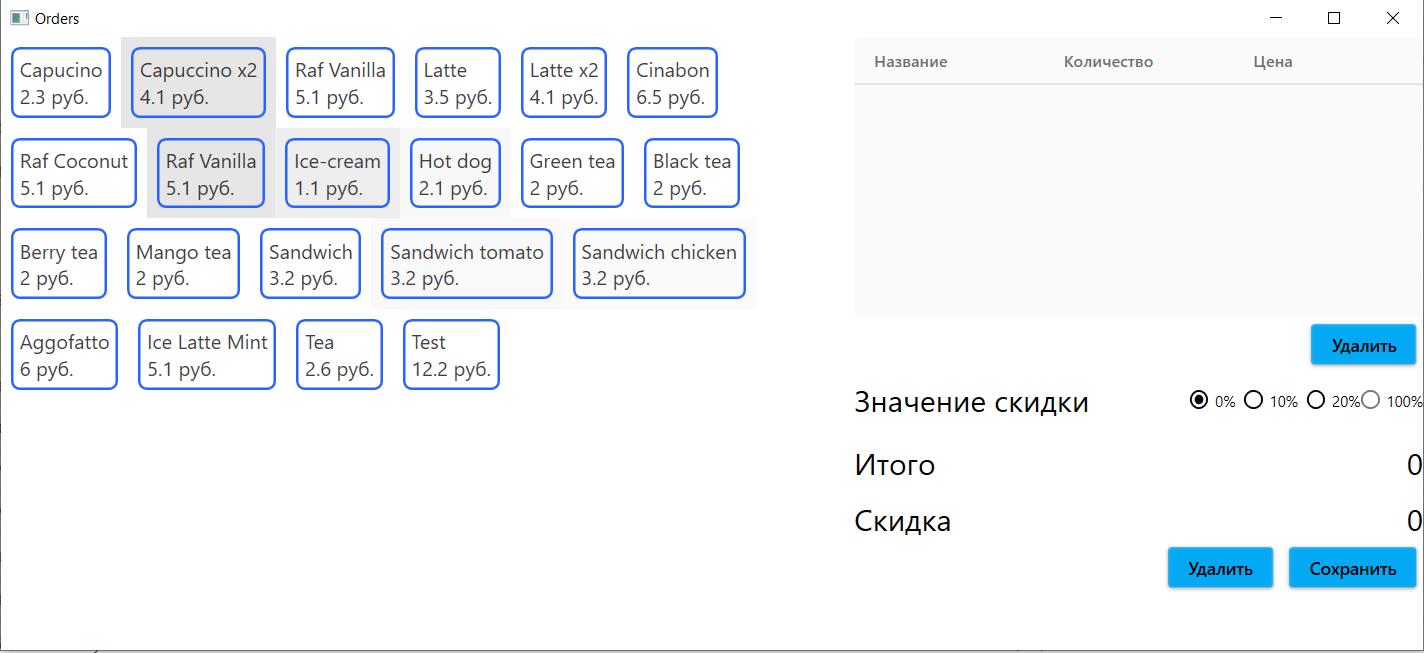


Рисунок 8.13 – кнопки продуктов

В окне редакции заказа (рисунок 8.14) есть возможность создать заказ или изменить его.

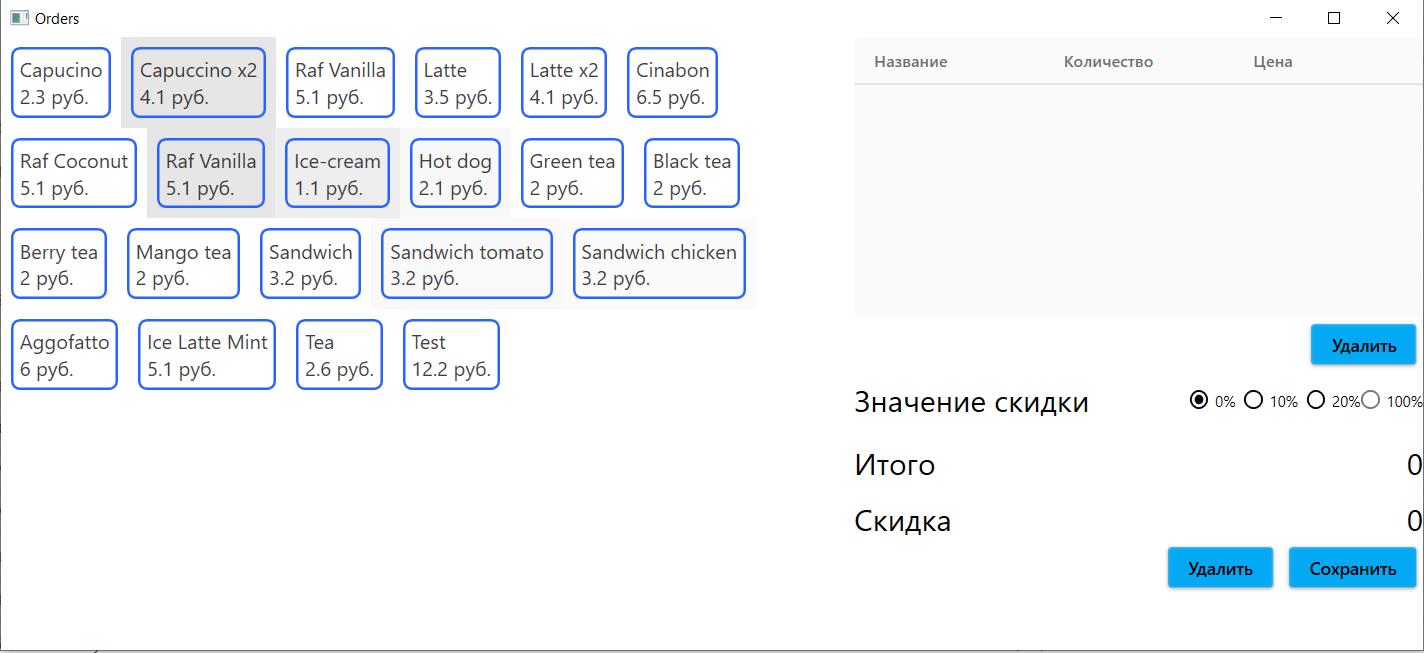


Рисунок 8.14 – окно создания заказа

В данном окне отображены все позиции, которые предлагает кофейня, а также есть возможность выставления скидки на заказ в размерах: 10, 20 и 100 процентов. Окно разделено на 2 части. Одна со всеми позициями, друга с редакцией информации о заказе, в котором виден список продуктов в заказе, выставленная скидка, сумма заказа. А также присутствуют кнопки для удаления и сохранения заказа и удаления продукта из заказа. Для того, чтобы добавить позицию в заказ, необходимо два раза щелкнуть по блюду, после чего оно давится в список блюд в заказе (рисунок 8.15).

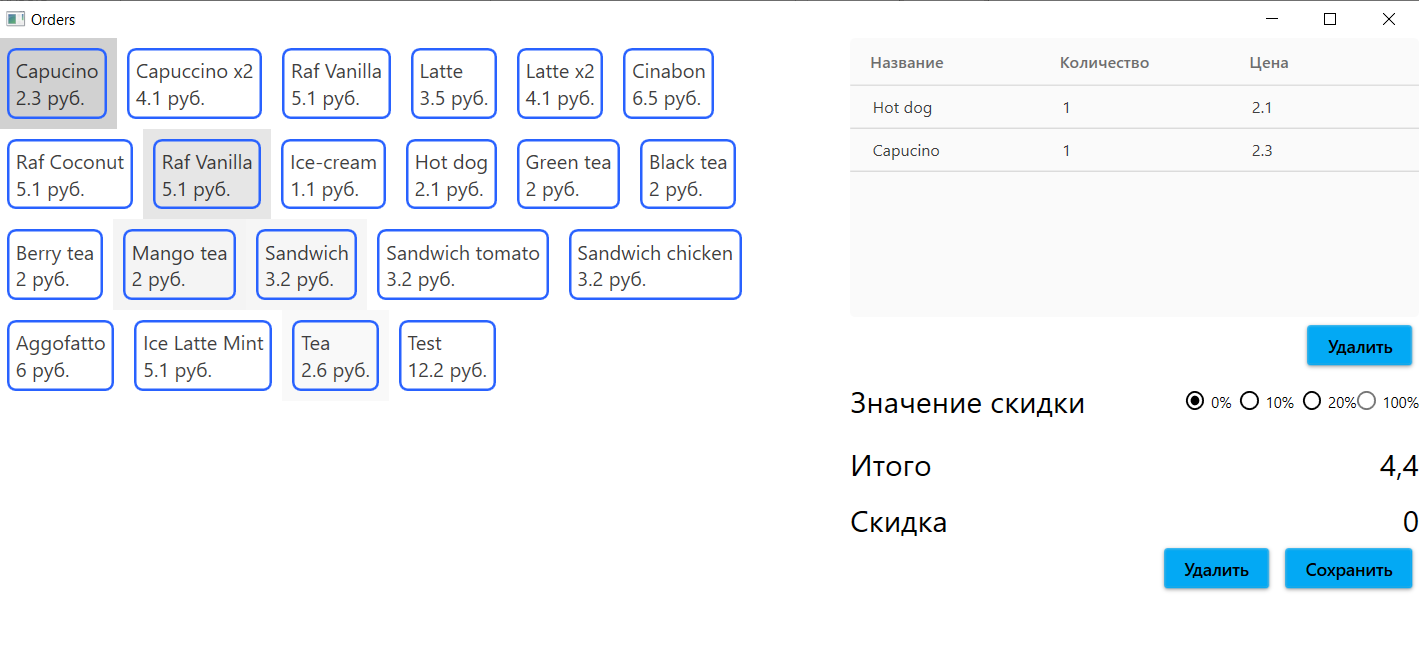


Рисунок 8.15 – Окно заказа после двойного нажатия на кнопки «Capucino» и «Hot dog»

Как видно из рисунка 8.14 в заказ были добавлены следующие продукты: «Capucino» и «Hot dog» для того, чтобы изменить количество продуктов, необходимо прямо в списке продуктов в заказе в колонке «Количество» изменить цифру 1 на любую другую (рисунок 8.16).

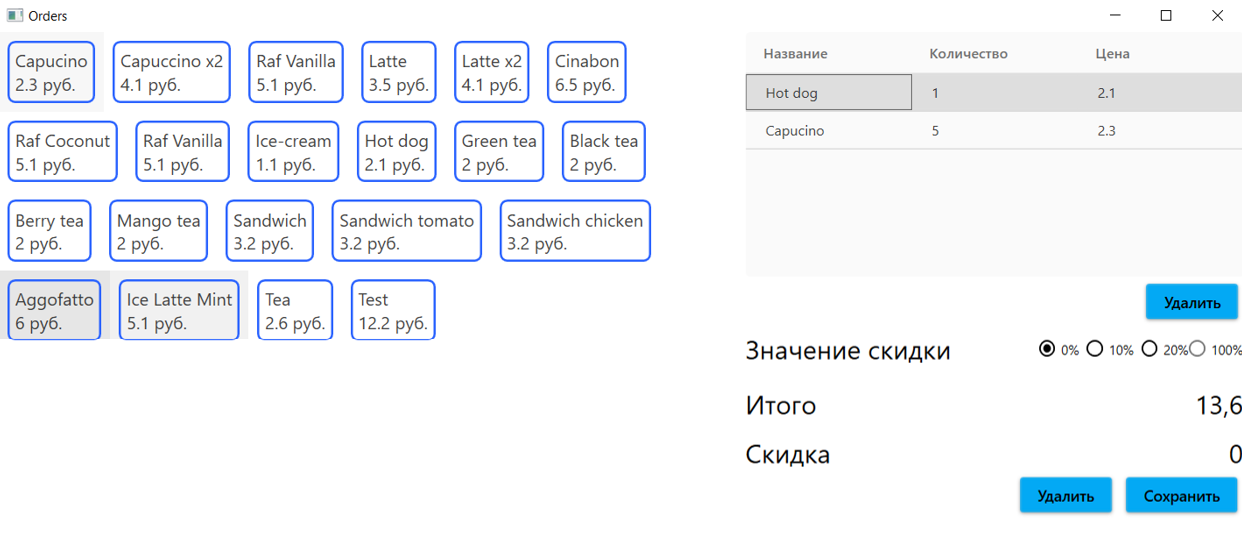


Рисунок 8.16 – Заказ с измененным количеством товаров

Если в заказе есть необходимость выставления скидки, то при клике на любую позицию с соответствующей скидкой в поле «Значение скидки» скидка будет просчитана и выставлена автоматически (рисунок 8.17).

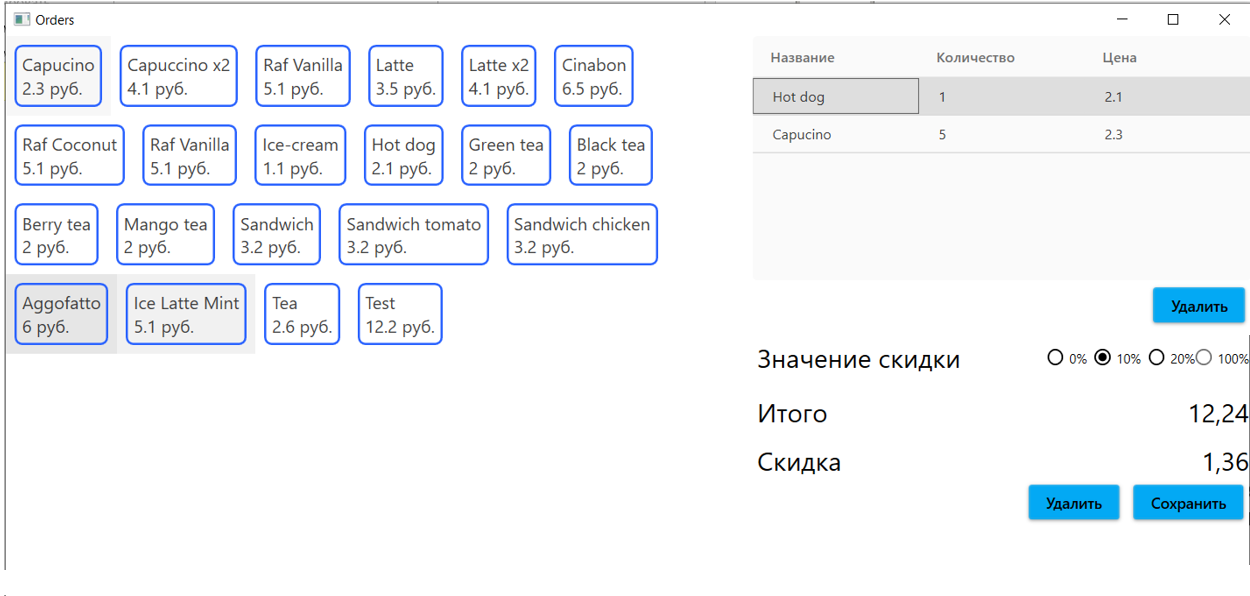


Рисунок 8.17 – Заказ с выставленной скидкой

После нажатия кнопки «Сохранить» заказ сохранится и появится в окне со списком всех заказов (рисунок 8.18)

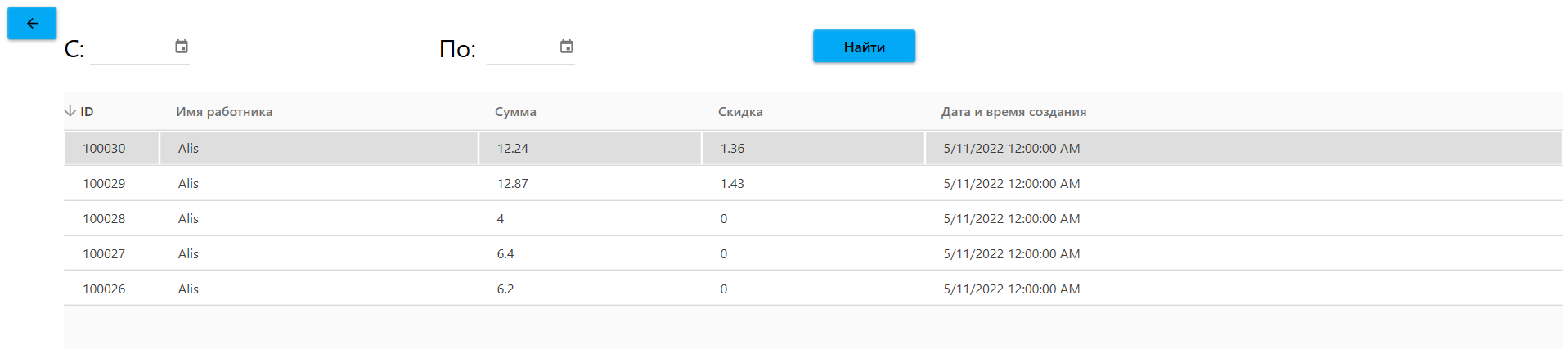


Рисунок 8.18 – Страница с заказами после сохранения нового заказа

Страница «Статистика» делает сводку по количеству заказов и выручки в определенный период (рисунок 8.19).

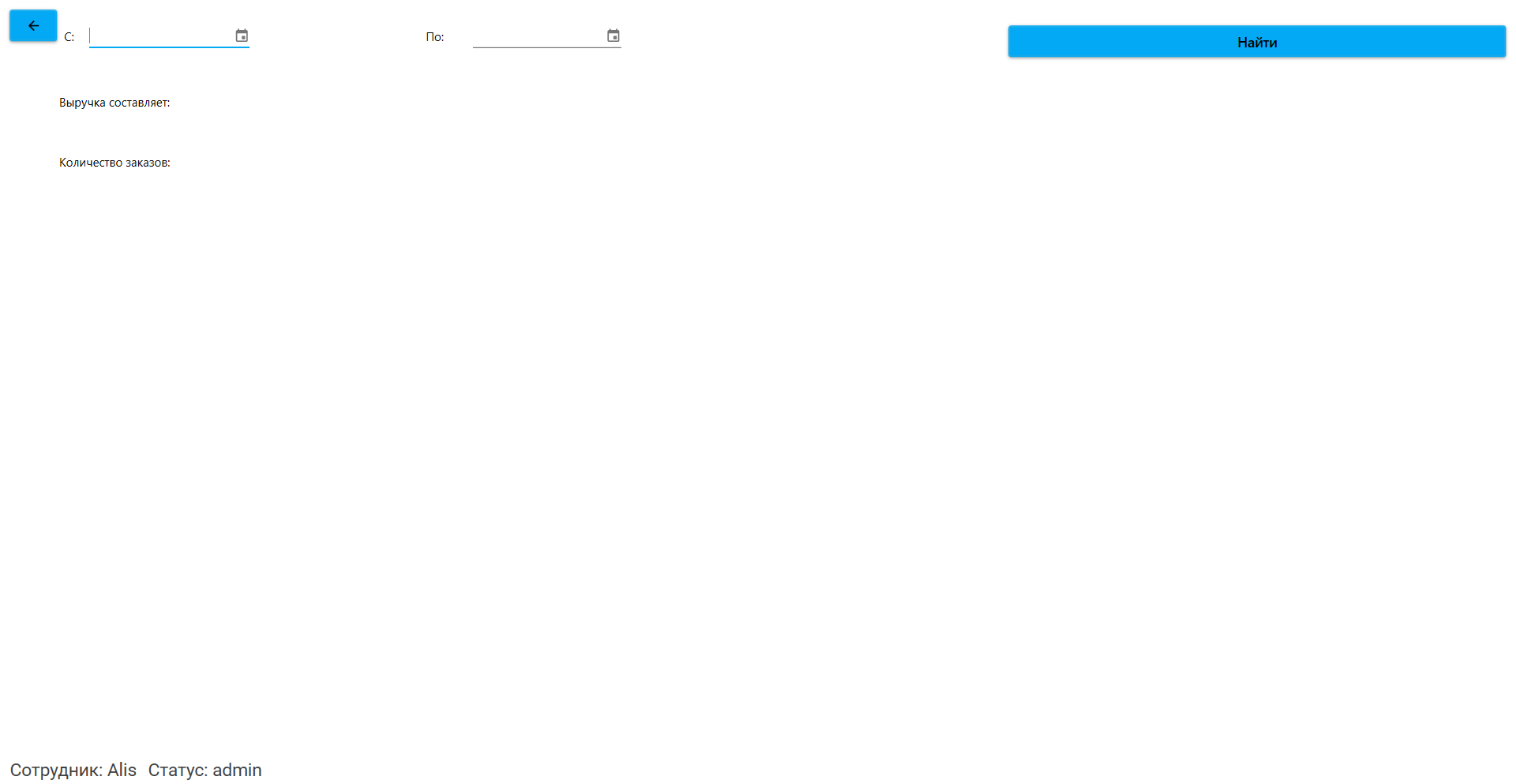


Рисунок 8.19 – Страница «Статистика»

Для того, чтобы получить статистику по определенному периоду, необходимо выбрать даты: с которой будут браться заказы и по какую. Далее необходимо нажать на кнопку «Найти» и появятся необходимые нам значения. На рисунке 8.20 представлена сводка с 18.04.2022 по 11.05.2022 год. Исходя из данной сводки мы можем сделать вывод что в период с 18.04.2022 по 11.05.2022 год у кофейни было сделано 1720 заказов, и общая сумма выручки составила 23733,29 руб.

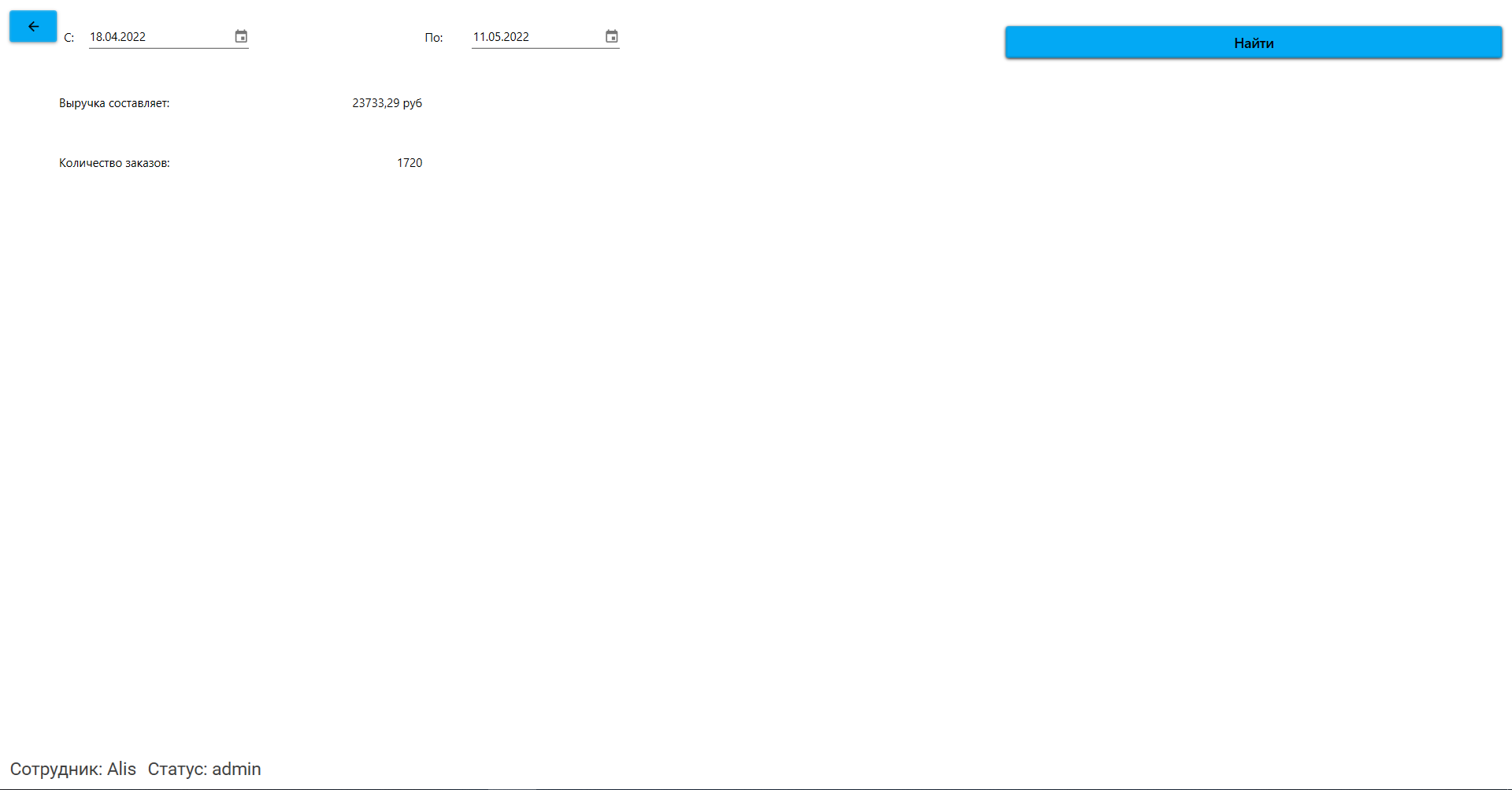


Рисунок 8.20 – Сводка данных за период с 18.04.2022 по 11.05.2022 год

После завершения работы пользователь может покинуть систему нажав на кнопку «Выход» в контекстном меню. После этого снова появится окно авторизации (рисунок 8.21).

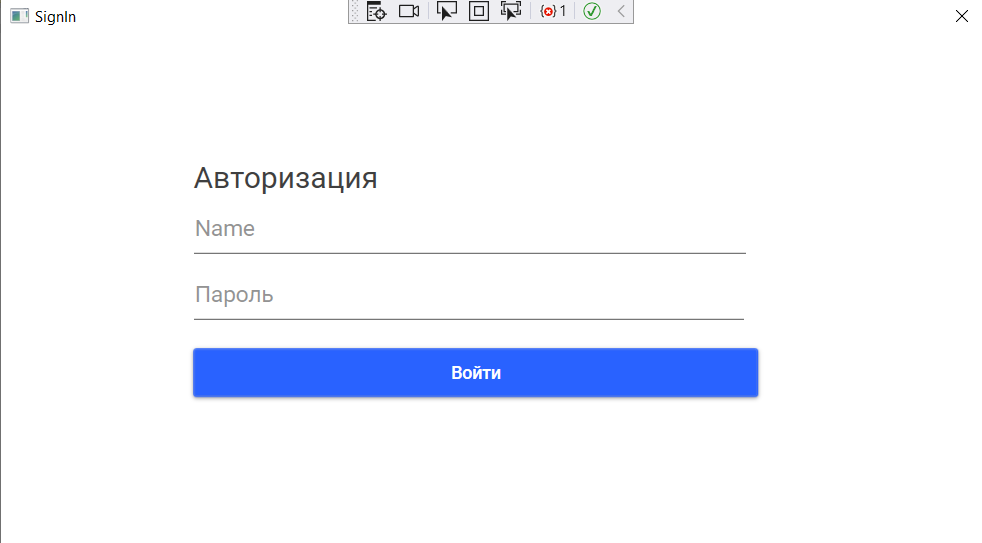


Рисунок 8.21 – Выход из системы

После завершения работы одного пользователя в систему может войти любой другой пользователь, у которого есть доступ.

На рисунке 8.22 представлено контекстное меню для обычного пользователя, ему доступна только таблица с заказами. Доступа к списку сотрудников, а также продуктов и их изменение или удаление, у него нет.



Рисунок 8.22 – Контекстное меню для пользователя с малыми привилегиями

# **Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных для автоматизированной системы «Кофейня», которая в совокупности с приложением формирует полноценное desktop-приложение для управления рабочим процессом. В данной работе использовалось СУБД Oracle DataBase 19c, а формирование desktop-приложения осуществлялось на языке C# с помощью системы для построения клиентских приложений Windows – WPF. При разработке курсового проекта использовались объекты: таблицы, хранимые процедуры, индексы, последовательности, пакеты.

Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением, которое помогло облегчить взаимодействие с базой данных посредством программного интерфейса. Взаимодействие между приложением и базой данных было реализовано с помощью технологии ADO.NET. При разработке выполнены следующие пункты:

* разработка приложения для взаимодействия с БД;
* регистрация/удаление сотрудника;
* изменение данных сотрудника;
* обновление списка фильмов;
* просмотр информации о каждом фильме;
* обновление списка сеансов;
* сортировка списка сеансов;
* продажа и возврат билетов;
* отправка билета по почте.

Приложение прошло тестирование при использовании в БД большого количество данных. Также были реализованы процедуры для импорта, экспорта данных в формат XML.

Была реализована технология «Резервное копирование и восстановление данных». Шифрование данных было реализовано с помощью пакета DBMS\_CRYPTO.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

# **Cписок источников**

1. Язык C# и .NET Framework [Электронный ресурс] – http://professorweb.ru/my/csharp/charp\_theory/level1/infonet.php.
2. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] – https://metanit.com/sharp/adonet/.
3. Документация Oracle [Электронный ресурс] / Foundation, Inc. https://docs.oracle.com/cd/B28359\_01/server.111/b31222/toc.htm
4. Официальный сайт Oracle [Электронный ресурс] / Foundation, Inc. https://www.oracle.com/database/database-vault/index.html
5. Продукты Oracle [Электронный ресурс] [Электронный ресурс] / Foundation, Inc. http://www.interface.ru/home.asp?artId=24678
6. Документы Oracle [Электронный ресурс] / Foundation, Inc. http://www.oracle.com/technetwork/database/security/database-vault-ds-12c-1898877.pdf

# **Приложение А**

//Создание таблиц

create table CaffeUser.Item

(

IdItem int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

Name varchar(50) not null,

Cost float not null,

IsDeleted CHAR(1) CHECK (IsDeleted IN ('1','0'))

)

tablespace data\_table;

drop table CaffeUser.Item;

create table CaffeUser.UserInCaffe

(

IdUser int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

Name varchar(50) not null,

Login varchar(50) not null,

Password varchar(50) not null,

Phone varchar(10) not null,

Created date not null,

DateOfBirth date not null,

Status varchar(50) default 'user' not null,

IsDeleted CHAR(1) CHECK (IsDeleted IN ('1','0'))

)

tablespace data\_table;

drop table CaffeUser.UserInCaffe;

CREATE TABLE CaffeUser.OrderInCaffe

(

IdOrder int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

IdUser int,

Created date default CURRENT\_DATE not null,

Discount float not null,

Cost float not null,

IsDeleted CHAR(1) CHECK (IsDeleted IN ('1','0')),

CONSTRAINT FK\_IdUser

FOREIGN KEY (IdUser)

REFERENCES CaffeUser.UserInCaffe(IdUser)

)

tablespace data\_table;

create table CaffeUser.ItemsInOrder

(

ItemsInOrder int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

IdOrder int,

IdItem int,

Count int not null,

CONSTRAINT FK\_IdOrder

FOREIGN KEY (IdOrder)

REFERENCES CaffeUser.OrderInCaffe(IdOrder),

CONSTRAINT FK\_IdItem

FOREIGN KEY (IdItem)

REFERENCES CaffeUser.Item(IdItem)

)

tablespace data\_table;

create table CaffeUser.StatusInCaffe

(

Status int primary key,

name varchar(50),

rang varchar(50)

)

tablespace data\_table;

# **Приложение Б**

--создание пользователей и придание им необходимых прав

CREATE BIGFILE TABLESPACE data\_table

DATAFILE 'data\_table1.dat'

SIZE 20M

AUTOEXTEND ON NEXT 10M MAXSIZE 1000M;

alter session set "\_ORACLE\_SCRIPT"=true;

CREATE USER CaffeUser

IDENTIFIED BY secret

DEFAULT TABLESPACE data\_table

QUOTA UNLIMITED on data\_table;

GRANT CREATE SESSION TO CaffeUser;

GRANT SELECT,INSERT,UPDATE,DELETE on CaffeUser.Item TO CaffeUser;

GRANT SELECT,INSERT,UPDATE,DELETE on CaffeUser.UserInCaffe TO CaffeUser;

GRANT SELECT,INSERT,UPDATE,DELETE on CaffeUser.OrderInCaffe TO CaffeUser;

GRANT SELECT,INSERT,UPDATE,DELETE on CaffeUser.ItemsInOrder TO CaffeUser;

alter session set "\_ORACLE\_SCRIPT"=true;

CREATE USER CaffeAdmin

IDENTIFIED BY secret

DEFAULT TABLESPACE data\_table

QUOTA UNLIMITED on data\_table;

GRANT CREATE SESSION TO CaffeRegularUser;

GRANT SELECT on CaffeUser.Item TO CaffeRegularUser;

GRANT SELECT,INSERT,UPDATE,DELETE on CaffeUser.OrderInCaffe TO CaffeRegularUser;

GRANT SELECT,INSERT,UPDATE,DELETE on CaffeUser.ItemsInOrder TO CaffeRegularUser;

# **Приложение В**

--Procedures

create or replace package CaffeUser.ItemNS is

type rowItem is record(

IdItem CaffeUser.Item.IdItem%TYPE,

Name CaffeUser.Item.Name%TYPE,

Cost CaffeUser.Item.Cost%TYPE,

IsDeleted CaffeUser.Item.IsDeleted%TYPE

);

type tblItem is table of rowItem;

--

function GetItem

(prm\_depID number default null)

return tblItem

pipelined;

function GetItemById

(prm\_depID number default null, id\_value int)

return tblItem

pipelined;

procedure InsertItem

( Name CaffeUser.Item.Name%TYPE,

Cost CaffeUser.Item.Cost%TYPE,

IsDeleted CaffeUser.Item.IsDeleted%TYPE);

procedure DeleteItem

(IdItem CaffeUser.Item.IdItem%TYPE);

procedure DeleteItemGeniusVersion

(IdItem\_t CaffeUser.Item.IdItem%TYPE);

procedure UpdateItem

( IdItem\_t CaffeUser.Item.IdItem%TYPE,

Name\_t CaffeUser.Item.Name%TYPE);

end ItemNS;

drop package CaffeUser.ItemNS;

create or replace package body CaffeUser.ItemNS is

function GetItem

(prm\_depID number default null)

return tblItem

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.Item

where IsDeleted = 0

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.Item

where IsDeleted = 0

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetItem;

function GetItemById

(prm\_depID number default null, id\_value int)

return tblItem

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.Item

where IdItem = id\_value

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.Item

where IdItem = id\_value

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetItemById;

procedure InsertItem

( Name CaffeUser.Item.Name%TYPE,

Cost CaffeUser.Item.Cost%TYPE,

IsDeleted CaffeUser.Item.IsDeleted%TYPE)

is

begin

insert into CaffeUser.Item(NAME, COST, ISDELETED)

values (NAME, COST, ISDELETED);

end InsertItem;

procedure UpdateItem

( IdItem\_t CaffeUser.Item.IdItem%TYPE,

Name\_t CaffeUser.Item.Name%TYPE)

is

begin

update CaffeUser.Item set Name = Name\_t where IdItem = IdItem\_t;

commit;

end UpdateItem;

procedure DeleteItem

(IdItem CaffeUser.Item.IdItem%TYPE)

is

begin

delete from CaffeUser.Item where CaffeUser.Item.IdItem = IdItem;

end DeleteItem;

procedure DeleteItemGeniusVersion

(IdItem\_t CaffeUser.Item.IdItem%TYPE)

is

begin

update CaffeUser.Item

set

IsDeleted = 1

where IdItem = IdItem\_t;

end;

end ItemNS;

exec CaffeUser.ItemNS.DeleteItemGeniusVersion(5);

commit;

SELECT \* FROM TABLE(CaffeUser.ItemNS.GetItem(1000));

----------------------------------------------------------------

create or replace package CaffeUser.UserInCaffeNS is

type rowUserInCaffe is record(

IdUser CaffeUser.UserInCaffe.IdUser%TYPE,

Name CaffeUser.UserInCaffe.Name%TYPE,

Login CaffeUser.UserInCaffe.Login%TYPE,

Password CaffeUser.UserInCaffe.Password%TYPE,

Phone CaffeUser.UserInCaffe.Phone%TYPE,

Created CaffeUser.UserInCaffe.Created%TYPE,

DateOfBirth CaffeUser.UserInCaffe.DateOfBirth%TYPE,

Status CaffeUser.UserInCaffe.Status%TYPE,

IsDeleted CaffeUser.UserInCaffe.IsDeleted%TYPE

);

type tblUserInCaffe is table of rowUserInCaffe;

--

function GetUserInCaffe

(prm\_depID number default null)

return tblUserInCaffe

pipelined;

function GetUserInCaffeByLogin

(prm\_depID number default null, Login\_t CaffeUser.UserInCaffe.Login%TYPE)

return tblUserInCaffe

pipelined;

function GetUserInCaffeById

(prm\_depID number default null, IdUser\_t CaffeUser.UserInCaffe.IdUser%TYPE)

return tblUserInCaffe

pipelined;

procedure InsertUserInCaffe

( Name CaffeUser.UserInCaffe.Name%TYPE,

Login CaffeUser.UserInCaffe.Login%TYPE,

Password CaffeUser.UserInCaffe.Password%TYPE,

Phone CaffeUser.UserInCaffe.Phone%TYPE,

Created string,

DateOfBirth string,

Status CaffeUser.UserInCaffe.Status%TYPE);

procedure DeleteUserInCaffe

(IdUser CaffeUser.UserInCaffe.IdUser%TYPE);

procedure DeleteUserInCaffeGeniusVersion

(IdUser\_t CaffeUser.UserInCaffe.IdUser%TYPE);

procedure UpdateUserInCaffe

( IdUser\_t CaffeUser.UserInCaffe.IdUser%TYPE,

Name\_t CaffeUser.UserInCaffe.Name%TYPE,

Login\_t CaffeUser.UserInCaffe.Login%TYPE,

Password\_t CaffeUser.UserInCaffe.Password%TYPE,

Phone\_t CaffeUser.UserInCaffe.Phone%TYPE,

DateOfBirth\_t string,

Status\_t CaffeUser.UserInCaffe.Status%TYPE);

end UserInCaffeNS;

drop package CaffeUser.UserInCaffeNS;

create or replace package body CaffeUser.UserInCaffeNS is

function GetUserInCaffe

(prm\_depID number default null)

return tblUserInCaffe

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.UserInCaffe

where IsDeleted = 0

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.UserInCaffe

where IsDeleted = 0

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetUserInCaffe;

function GetUserInCaffeByLogin

(prm\_depID number default null, Login\_t CaffeUser.UserInCaffe.Login%TYPE)

return tblUserInCaffe

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.UserInCaffe

where login = Login\_t

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.UserInCaffe

where login = Login\_t

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetUserInCaffeByLogin;

function GetUserInCaffeById

(prm\_depID number default null, IdUser\_t CaffeUser.UserInCaffe.IdUser%TYPE)

return tblUserInCaffe

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.UserInCaffe

where iduser = IdUser\_t

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.UserInCaffe

where iduser = IdUser\_t

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetUserInCaffeById;

procedure InsertUserInCaffe

( Name CaffeUser.UserInCaffe.Name%TYPE,

Login CaffeUser.UserInCaffe.Login%TYPE,

Password CaffeUser.UserInCaffe.Password%TYPE,

Phone CaffeUser.UserInCaffe.Phone%TYPE,

Created string,

DateOfBirth string,

Status CaffeUser.UserInCaffe.Status%TYPE)

is

begin

insert into CaffeUser.UserInCaffe(Name, Login, Password, Phone, Created, DateOfBirth, Status, IsDeleted)

values (Name, Login, Password, Phone, TO\_DATE(Created, 'dd.mm.yyyy'), TO\_DATE(DateOfBirth, 'dd.mm.yyyy'), Status, 0);

end InsertUserInCaffe;

procedure UpdateUserInCaffe

( IdUser\_t CaffeUser.UserInCaffe.IdUser%TYPE,

Name\_t CaffeUser.UserInCaffe.Name%TYPE,

Login\_t CaffeUser.UserInCaffe.Login%TYPE,

Password\_t CaffeUser.UserInCaffe.Password%TYPE,

Phone\_t CaffeUser.UserInCaffe.Phone%TYPE,

DateOfBirth\_t string,

Status\_t CaffeUser.UserInCaffe.Status%TYPE)

is

begin

update CaffeUser.UserInCaffe set Name = Name\_t, Login = Login\_t, Password = Password\_t,

Phone = Phone\_t, DateOfBirth = to\_date(DateOfBirth\_t, 'dd.mm.yyyy'), Status = Status\_t

where IdUser = IdUser\_t;

end UpdateUserInCaffe;

procedure DeleteUserInCaffe

(IdUser CaffeUser.UserInCaffe.IdUser%TYPE)

is

begin

delete from CaffeUser.UserInCaffe where CaffeUser.UserInCaffe.IdUser = IdUser;

end DeleteUserInCaffe;

procedure DeleteUserInCaffeGeniusVersion

(IdUser\_t CaffeUser.UserInCaffe.IdUser%TYPE)

is

begin

update CaffeUser.UserInCaffe set IsDeleted = 1

where IdUser = IdUser\_t;

end DeleteUserInCaffeGeniusVersion;

end UserInCaffeNS;

-----------------------------------------------------------------

create or replace package CaffeUser.OrderInCaffeNS is

type rowOrderInCaffe is record(

IdOrder CaffeUser.OrderInCaffe.IdOrder%TYPE,

IdUser CaffeUser.OrderInCaffe.IdUser%TYPE,

Created CaffeUser.OrderInCaffe.Created%TYPE,

Discount CaffeUser.OrderInCaffe.Discount%TYPE,

IsDeleted CaffeUser.OrderInCaffe.IsDeleted%TYPE

);

type tblOrderInCaffe is table of rowOrderInCaffe;

--

function GetOrderInCaffe

(prm\_depID number default null)

return tblOrderInCaffe

pipelined;

function GetOrderInCaffeById

(prm\_depID number default null, id\_t int)

return tblOrderInCaffe

pipelined;

function GetOrderInCaffeFromDate

(prm\_depID number default null, from\_date string, to\_date\_ string)

return tblOrderInCaffe

pipelined;

function GetCountInInterval(from\_date string, to\_date\_ string)

return int;

function GetCountOfAllOrder

return int;

procedure InsertOrderInCaffe

( IdUser CaffeUser.OrderInCaffe.IdUser%TYPE,

Created string,

Discount CaffeUser.OrderInCaffe.Discount%TYPE,

IsDeleted CaffeUser.OrderInCaffe.IsDeleted%TYPE);

procedure DeleteOrderInCaffe

(IdOrder CaffeUser.OrderInCaffe.IdOrder%TYPE);

procedure DeleteOrderInCaffeGeniusVersion

(IdOrder\_t CaffeUser.OrderInCaffe.IdOrder%TYPE);

procedure UpdateOrderInCaffe

( IdOrder\_t CaffeUser.OrderInCaffe.IdOrder%TYPE,

IdUser\_t CaffeUser.OrderInCaffe.IdUser%TYPE,

Discount\_t CaffeUser.OrderInCaffe.Discount%TYPE);

end OrderInCaffeNS;

drop package CaffeUser.OrderInCaffeNS;

create or replace package body CaffeUser.OrderInCaffeNS is

function GetOrderInCaffe

(prm\_depID number default null)

return tblOrderInCaffe

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.OrderInCaffe

where isdeleted = 0

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.OrderInCaffe

where isdeleted = 0

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetOrderInCaffe;

function GetOrderInCaffeById

(prm\_depID number default null, id\_t int)

return tblOrderInCaffe

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.OrderInCaffe

where IdOrder = id\_t

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.OrderInCaffe

where IdOrder = id\_t

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetOrderInCaffeById;

function GetOrderInCaffeFromDate

(prm\_depID number default null, from\_date string, to\_date\_ string)

return tblOrderInCaffe

pipelined

is

from\_date\_t date;

to\_date\_t date;

begin

if from\_date is null and to\_date\_ is null then

from\_date\_t := to\_date('01.01.1970', 'dd.mm.yyyy');

to\_date\_t := current\_date;

end if;

if from\_date is null and to\_date\_ is not null then

from\_date\_t := to\_date('01.01.1970', 'dd.mm.yyyy');

to\_date\_t := to\_date(to\_date\_, 'dd.mm.yyyy');

end if;

if from\_date is not null and to\_date\_ is null then

from\_date\_t := to\_date(from\_date, 'dd.mm.yyyy');

to\_date\_t := current\_date;

end if;

if from\_date is not null and to\_date\_ is not null then

from\_date\_t := to\_date(from\_date, 'dd.mm.yyyy');

to\_date\_t := to\_date(to\_date\_, 'dd.mm.yyyy');

end if;

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.OrderInCaffe

where Created >= from\_date\_t and Created <= to\_date\_t and isdeleted = 0

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.OrderInCaffe

where Created >= from\_date\_t and Created <= to\_date\_t and isdeleted = 0

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end;

function GetCountInInterval(from\_date string, to\_date\_ string)

return int

is

return\_value int;

from\_date\_t date;

to\_date\_t date;

begin

if from\_date is null and to\_date\_ is null then

from\_date\_t := to\_date('01.01.1970', 'dd.mm.yyyy');

to\_date\_t := current\_date;

end if;

if from\_date is null and to\_date\_ is not null then

from\_date\_t := to\_date('01.01.1970', 'dd.mm.yyyy');

to\_date\_t := to\_date(to\_date\_, 'dd.mm.yyyy');

end if;

if from\_date is not null and to\_date\_ is null then

from\_date\_t := to\_date(from\_date, 'dd.mm.yyyy');

to\_date\_t := current\_date;

end if;

if from\_date is not null and to\_date\_ is not null then

from\_date\_t := to\_date(from\_date, 'dd.mm.yyyy');

to\_date\_t := to\_date(to\_date\_, 'dd.mm.yyyy');

end if;

select count(\*) into return\_value from CaffeUser.OrderInCaffe

where created >= from\_date\_t and created <= to\_date\_t;

return return\_value;

end;

function GetCountOfAllOrder

return int

is

return\_value int;

begin

select count(\*) into return\_value from CaffeUser.OrderInCaffe;

return return\_value;

end;

procedure InsertOrderInCaffe

( IdUser CaffeUser.OrderInCaffe.IdUser%TYPE,

Created string,

Discount CaffeUser.OrderInCaffe.Discount%TYPE,

IsDeleted CaffeUser.OrderInCaffe.IsDeleted%TYPE)

is

begin

insert into CaffeUser.OrderInCaffe(IdUser, created, discount, isdeleted)

values (IdUser, to\_date(Created, 'dd.mm.yyyy'), Discount, IsDeleted);

end InsertOrderInCaffe;

procedure UpdateOrderInCaffe

( IdOrder\_t CaffeUser.OrderInCaffe.IdOrder%TYPE,

IdUser\_t CaffeUser.OrderInCaffe.IdUser%TYPE,

Discount\_t CaffeUser.OrderInCaffe.Discount%TYPE)

is

begin

update CaffeUser.OrderInCaffe set IdUser = IdUser\_t,

Discount = Discount\_t

where IdOrder = IdOrder\_t;

end UpdateOrderInCaffe;

procedure DeleteOrderInCaffe

(IdOrder CaffeUser.OrderInCaffe.IdOrder%TYPE)

is

begin

delete from CaffeUser.OrderInCaffe

where CaffeUser.OrderInCaffe.IdOrder = IdOrder;

end DeleteOrderInCaffe;

procedure DeleteOrderInCaffeGeniusVersion

(IdOrder\_t CaffeUser.OrderInCaffe.IdOrder%TYPE)

is

begin

update CaffeUser.OrderInCaffe set IsDeleted = 1

where IdOrder\_t = IdOrder;

end;

end OrderInCaffeNS;

select CaffeUser.OrderInCaffeNS.GetCountOfAllOrder from dual;

select \* from CaffeUser.OrderInCaffe where idorder = 100028

select \* from CaffeUser.ItemsInOrder where itemsinorder = 643267

-----------------------------------------------------------------

create or replace package CaffeUser.ItemsInOrderNS is

type rowItemsInOrder is record(

ItemsInOrder CaffeUser.ItemsInOrder.ItemsInOrder%TYPE,

IdOrder CaffeUser.ItemsInOrder.IdOrder%TYPE,

IdItem CaffeUser.ItemsInOrder.IdItem%TYPE,

Count CaffeUser.ItemsInOrder.Count%TYPE

);

type tblItemsInOrder is table of rowItemsInOrder;

--

function GetItemsInOrder

(prm\_depID number default null)

return tblItemsInOrder

pipelined;

function GetListOfItemsInOrderById

(prm\_depID number default null, IdOrder\_t CaffeUser.ItemsInOrder.IdOrder%TYPE)

return tblItemsInOrder

pipelined;

procedure InsertItemsInOrder

( IdOrder CaffeUser.ItemsInOrder.IdOrder%TYPE,

IdItem CaffeUser.ItemsInOrder.IdItem%TYPE,

Count CaffeUser.ItemsInOrder.Count%TYPE);

procedure DeleteItemsInOrder

(ItemsInOrder\_ CaffeUser.ItemsInOrder.ItemsInOrder%TYPE);

procedure UpdateItemsInOrder

( ItemsInOrder\_t CaffeUser.ItemsInOrder.ItemsInOrder%TYPE,

IdOrder\_t CaffeUser.ItemsInOrder.IdOrder%TYPE,

IdItem\_t CaffeUser.ItemsInOrder.IdItem%TYPE,

Count\_t CaffeUser.ItemsInOrder.Count%TYPE);

procedure UpdateItemsInOrderOfCount

( ItemsInOrder\_t CaffeUser.ItemsInOrder.ItemsInOrder%TYPE,

Count\_t CaffeUser.ItemsInOrder.Count%TYPE);

end ItemsInOrderNS;

drop package CaffeUser.ItemsInOrderNS;

create or replace package body CaffeUser.ItemsInOrderNS is

function GetItemsInOrder

(prm\_depID number default null)

return tblItemsInOrder

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.ItemsInOrder

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.ItemsInOrder

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetItemsInOrder;

function GetListOfItemsInOrderById

(prm\_depID number default null, IdOrder\_t CaffeUser.ItemsInOrder.IdOrder%TYPE)

return tblItemsInOrder

pipelined

is

begin

if prm\_depID is null then

for curr in

(

select \*

from CaffeUser.ItemsInOrder

where IdOrder = IdOrder\_t

) loop

pipe row (curr);

end loop;

else

for curr in

(

select \*

from CaffeUser.ItemsInOrder

where IdOrder = IdOrder\_t

FETCH FIRST prm\_depID ROWS ONLY

) loop

pipe row (curr);

end loop;

end if;

end GetListOfItemsInOrderById;

procedure InsertItemsInOrder

( IdOrder CaffeUser.ItemsInOrder.IdOrder%TYPE,

IdItem CaffeUser.ItemsInOrder.IdItem%TYPE,

Count CaffeUser.ItemsInOrder.Count%TYPE)

is

begin

insert into CaffeUser.ItemsInOrder(IdOrder, IdItem, Count)

values (IdOrder, IdItem, Count);

end InsertItemsInOrder;

procedure UpdateItemsInOrder

( ItemsInOrder\_t CaffeUser.ItemsInOrder.ItemsInOrder%TYPE,

IdOrder\_t CaffeUser.ItemsInOrder.IdOrder%TYPE,

IdItem\_t CaffeUser.ItemsInOrder.IdItem%TYPE,

Count\_t CaffeUser.ItemsInOrder.Count%TYPE)

is

begin

update CaffeUser.ItemsInOrder set IdOrder = IdOrder\_t, IdItem = IdItem\_t,

Count = Count\_t

where ItemsInOrder = ItemsInOrder\_t;

end UpdateItemsInOrder;

procedure UpdateItemsInOrderOfCount

( ItemsInOrder\_t CaffeUser.ItemsInOrder.ItemsInOrder%TYPE,

Count\_t CaffeUser.ItemsInOrder.Count%TYPE)

is

begin

update CaffeUser.ItemsInOrder set Count = Count\_t

where ItemsInOrder = ItemsInOrder\_t;

end UpdateItemsInOrderOfCount;

procedure DeleteItemsInOrder

(ItemsInOrder\_ CaffeUser.ItemsInOrder.ItemsInOrder%TYPE)

is

begin

delete from CaffeUser.ItemsInOrder

where ItemsInOrder = ItemsInOrder\_;

end DeleteItemsInOrder;

end ItemsInOrderNS;

create or replace function GetSumCostFromInterval(date1 date, date2 date)

return float

is

return\_value int;

begin

end;

# **Приложение Г**

--XML

create or replace package body helpers as

procedure clob\_to\_file(p\_clob in clob,

p\_dir in varchar2,

p\_filename in varchar2)

as

l\_file utl\_file.file\_type;

l\_buffer varchar2(32767);

l\_amount binary\_integer := 32767;

l\_pos integer := 1;

begin

l\_file := utl\_file.fopen(p\_dir, p\_filename, 'w', 32767);

loop

dbms\_lob.read(p\_clob, l\_amount, l\_pos, l\_buffer);

utl\_file.put(l\_file, l\_buffer);

utl\_file.fflush(l\_file);

l\_pos := l\_pos + l\_amount;

end loop;

exception

when no\_data\_found then

if utl\_file.is\_open(l\_file) then

utl\_file.fclose(l\_file);

end if;

when others then

if utl\_file.is\_open(l\_file) then

utl\_file.fclose(l\_file);

end if;

raise;

end clob\_to\_file;

function load\_clob\_from\_file(p\_directory in varchar2, p\_file\_name in varchar2) return clob

as

l\_clob clob;

l\_bfile bfile;

dst\_offset number := 1 ;

src\_offset number := 1 ;

lang\_ctx number := dbms\_lob.default\_lang\_ctx;

warning number;

begin

l\_bfile := bfilename(p\_directory, p\_file\_name);

dbms\_lob.fileopen(l\_bfile);

dbms\_lob.createtemporary(l\_clob, true);

dbms\_lob.loadclobfromfile(

dest\_lob => l\_clob,

src\_bfile => l\_bfile,

amount => dbms\_lob.getlength(l\_bfile),

dest\_offset => dst\_offset,

src\_offset => src\_offset,

bfile\_csid => dbms\_lob.default\_csid,

lang\_context => lang\_ctx,

warning => warning);

dbms\_lob.fileclose(l\_bfile);

return l\_clob;

end load\_clob\_from\_file;

end helpers;

create or replace package body io as

procedure import (p\_directory varchar2, p\_filename varchar2) as

l\_clob clob;

l\_xml xmltype;

begin

l\_clob := helpers.load\_clob\_from\_file(p\_directory, p\_filename);

l\_xml := xmltype(l\_clob);

insert into faculties select xml\_tab.\* from dual, xmltable ('//faculty' passing l\_xml columns

id number path '@id',

name nvarchar2(70) path 'text()'

) xml\_tab;

insert into departments select xml\_tab.\* from dual, xmltable ('//department' passing l\_xml columns

id number path '@id',

name nvarchar2(70) path 'text()',

stack\_id number path './../@id',

short\_name nvarchar2(20) path 'short\_name'

) xml\_tab;

insert into subjects select xml\_tab.\* from dual, xmltable ('//subject' passing l\_xml columns

id number path '@id',

name nvarchar2(70) path 'text()',

department\_id number path './../@id'

) xml\_tab;

end import;

procedure export(p\_directory varchar2, p\_filename varchar2) as

l\_clob clob;

begin

select xmltype.getclobval(xml\_view.faculties) into l\_clob from xml\_view;

helpers.clob\_to\_file(l\_clob, p\_directory, p\_filename);

end export;

end io;

--------------------------------------------------------

-- DDL for View XML\_VIEW

--------------------------------------------------------

CREATE OR REPLACE FORCE VIEW "ORDSTESTUSER"."XML\_VIEW" ("FACULTIES") AS

with xml\_items(elements) as (

select xmlelement("item",

xmlattributes(

IDITEM as "id",

isdeleted as "deleted"

),

name,

xmlelement(

"cost", cost

))

from CaffeUser.Item fetch first 10 rows only

),

xml\_orders(elements, order\_id) as (

select xmlelement("order",

xmlattributes(

itemsinorder as "id",

idorder as "idorder"

),

xmlforest(

iditem as "iditem",

count as "count"

)

), idorder from CaffeUser.ItemsInOrder fetch first 10 rows only

)

select xmlelement("items", xmlforest(

(select xmlagg(elements) from xml\_orders) as "orders",

(select xmlagg(elements) from xml\_items) as "items"

)) as items from dual;