|  |
| --- |
| 09.02.07 ПР-419  **ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**  **ПМ.11 Разработка, администрирование и защита баз данных** |
| **УП.11.02 Разработка приложения** |

|  |  |  |  |
| --- | --- | --- | --- |
| Практикант |  |  | Мун А. О. |
| Руководитель практики |  |  | **Пьяненкова А. В.** |
|  |  |  |  |

Оглавление

[УП.11.02 Разработка приложения 3](#_Toc118539666)

[**База практики** 3](#_Toc118539667)

[**Период прохождения практики** 3](#_Toc118539668)

[**Основная задача практики** 3](#_Toc118539669)

[**Виды работ** 3](#_Toc118539670)

[**Задание на разработку** 4](#_Toc118539671)

[Описание организации данных 6](#_Toc118539672)

[**Логическая модель данных** 6](#_Toc118539673)

[**Схема данных** 7](#_Toc118539674)

[**Описание таблиц** 7](#_Toc118539675)

[**Описание интерфейса** 10](#_Toc118539676)

[**Описание отчетов** 17](#_Toc118539677)

[Заключение 19](#_Toc118539678)

[Используемые источники информации 20](#_Toc118539679)

**УП.11.02 Разработка приложения**

**База практики**

Практика проводилась на базе полигона вычислительной техники БПОУ ОО «Омский авиационный колледж имени Н.Е. Жуковского»

**Период прохождения практики**

Практика Разработка приложения по разработке программных продуктов проходила с **31.10.2022 г.** по 0**5.11.2022 г.** на полигоне вычислительной техники Омавиат.

**Основная задача практики**

Основная задача практики - формирование практических навыков по разработке кода программного продукта на основе готовой спецификации на уровне модуля, проведения тестирования программного модуля по определенному сценарию, использования основных методологий процессов разработки программного обеспечения, работы с объектами базы данных в конкретной СУБД.

**Виды работ**

* создание базы данных;
* разграничение данных пользователей;
* разработка приложения;
* документирование и презентация решений/

**Задание на разработку**

**Описание**

Требуется разработать программное решение для учета платежей физических лиц. Используя полученный программный продукт, физические лица могут вести учет своих платежей. Кроме того, появится возможность производить несложный анализ затрат в разрезе периодов или категорий. Также можно получать результаты анализа в печатной форме.

**Требования**

**Разграничение данных пользователей**

1. Программное решение должно быть многопользовательским;
2. Пароли должны обязательно храниться в зашифрованном виде;
3. Логин можно выбирать из выпадающего списка;

**Окно с основными данными**

1. В окне основных данных выводятся только данные текущего пользователя;
2. Данные должны помещаться на один экран по ширине;
3. Данные можно отображать за выбранный период (от даты до даты);
4. Данные можно отображать по одной из категорий;
5. Данные можно добавлять;
6. Данные можно удалять;
7. По отображаемым данным можно получить отчет;

**Добавление платежей**

1. При добавлении данных можно указывать категорию платежа;
2. При добавлении данных обязательно указывается назначение платежа;
3. Назначение платежа указывается на русском языке;
4. Назначение платежа состоит как минимум из 3 букв;
5. Едино разово можно совершить несколько одинаковых платежей;
6. Количество обязательно должно быть целым положительным числом;
7. Для платежа обязательно указывается цена в рублях;
8. Цена не может быть отрицательной;
9. Стоимость платежа рассчитывается как количество \* цена;
10. Дата платежа указывается как текущее время на компьютере;

**Удаление платежей**

1. При удалении данных система должна запрашивать подтверждение;
2. В подтверждении нужно выводить назначение платежа и другую значимую информацию;
3. Удаление должно сопровождаться звуковым или визуальным оформлением;

**Отчет**

1. Отчет может выводиться встроенными средствами, либо в распространенных программных продуктах (Microsoft Office Word, Microsoft Office Excel, Internet-браузер);
2. У отчета обязательно должен быть заголовок на первой странице;
3. В заголовке может быть указан период и категория платежей;
4. На каждой странице указывается ФИО пользователя и номер страницы;
5. Платежи группируются по категориям;
6. Внутри каждой категории платежи сортируются по дате;
7. В конце отчета выводится суммарная стоимость всех показанных платежей.

Описание организации данных

**Логическая модель данных**

Таблица 1. Описание основных объектов разрабатываемой системы

|  |  |
| --- | --- |
| Объект | Описание |
| Users | Информация о пользователях системы (ФИО, логин и пароль) |
| Products | Информация о продуктах (Наименование) |
| Payments | Информация о платежах (Количество, Сумма, Цена, Дата, Назначение, Чек) |
| Categories | Информация о категориях (Наименование) |
| Analyzes | Информация о изменениях платежей за определенный день |

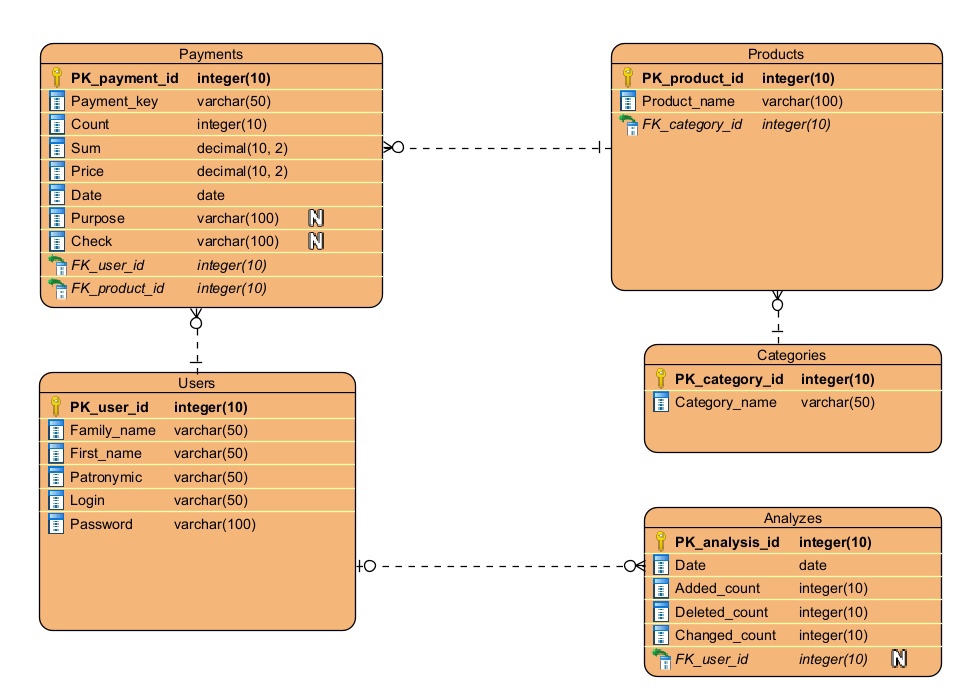


Рисунок 1 - ER-диаграмма

**Схема данных**

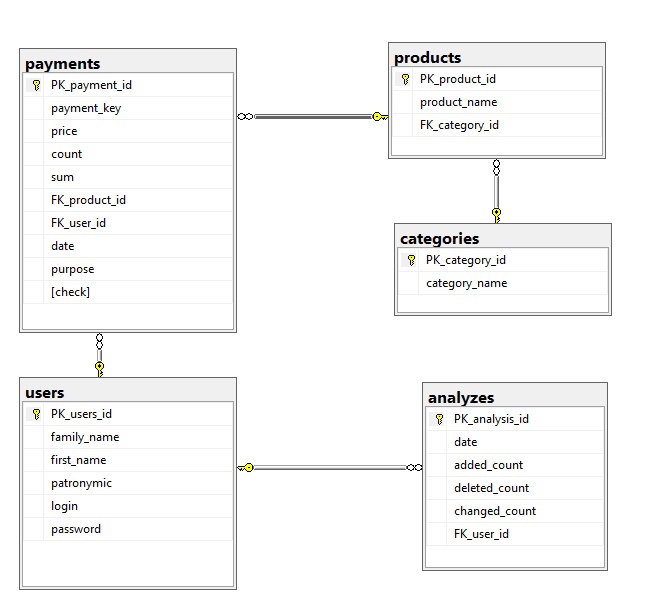


Рисунок 2 - Физическая модель данных

**Описание таблиц**

Таблица 2. Users

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_user\_id | Ключевое поле таблицы users | Integer | 10 |
| Family\_name | Фамилия пользователя | Varchar | 50 |
| First\_name | Имя пользователя | Varchar | 50 |
| Patronymic | Отчество пользователя | Varchar | 50 |
| Login | Логин пользователя | Varchar | 50 |
| Password | Пароль пользователя | Varchar | 100 |

*Таблица 3. Products*

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_product\_id | Ключевое поле таблицы Products | Integer | 10 |
| Product\_name | Наименование продукта | Varchar | 100 |
| FK\_category\_id | Внешний ключ таблицы Products к таблице Categories | Integer | 10 |

*Таблица 4. Payments*

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_payment\_id | Ключевое поле таблицы payments | Integer | 10 |
| Payment\_key | Уникальный ключ платежа | Varchar | 50 |
| Count | Количество продуктов | Integer | 10 |
| Sum | Итоговая цена (Count \* Price) | Decimal | 10,2 |
| Price | Цена продукта | Decimal | 10,2 |
| Data | Дата совершения платежа | Date | - |
| Purpose | Назначение платежа | Varchar | 100 |
| Check | Путь до изображения чека | Varchar | 100 |
| FK\_user\_id | Внешний ключ таблицы Payments к таблице Users | Integer | 10 |
| FK\_product\_id | Внешний ключ таблицы Payments к таблице Products | Integer | 10 |

*Таблица 5. Categories*

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_category\_id | Ключевое поле таблицы Categories | Integer | 10 |
| Category\_name | Наименование категории | Varchar | 50 |

*Таблица 6. Analyzes*

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_analysis\_id | Ключевое поле таблицы Analyzes | Integer | 10 |
| Date | Дата изменений | Date | - |
| Added\_count | Количество добавленных изменений | Integer | 10 |
| Deleted\_count | Количество добавленных удалений | Integer | 10 |
| Changed\_count | Количество добавленных изменений | Integer | 10 |
| FK\_user\_id | Внешний ключ таблицы Analyzes к таблице users | Integer | 10 |

**Описание интерфейса**

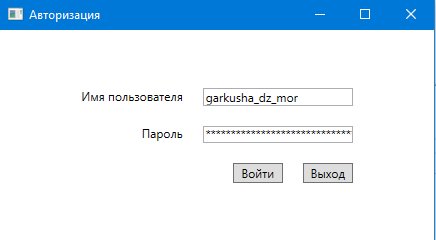
**

Рисунок 3 - Окно авторизации

На форме авторизации отображены два текстовых поля для ввода имени и пароля пользователя, валидирующихся на корректность ввода.

При нажатии на кнопку Выход, окно авторизации закрывается.

При нажатии на кнопку Войти происходит авторизация пользователя, при неудачной авторизации перед пользователям появляется окно с каптчей.

В случае 3 неверных попыток авторизации, окно блокируется на 30 секунд.

Листинг 1 - листинг кода окна авторизации

|  |
| --- |
| if (!string.IsNullOrEmpty(textBoxName.Text) && !string.IsNullOrEmpty(textBoxPassword.Password))  {  var user = Instances.db.users.Where(p => p.login == textBoxName.Text && p.password == textBoxPassword.Password).FirstOrDefault();  if (user != null && Hashing.hashSHA256(user.login).ToLower()==user.password)  {  MessageBox.Show("Успешная авторизация");  Window windowPayment = new WindowPayments(user);  windowPayment.Show();  Close(); }  Else {  MessageBox.Show("Неверные данные для авторизации");  new CaptchaWindow().ShowDialog();  tries--;  if (tries <= 0) {  textBoxName.IsEnabled = false;  textBoxPassword.IsEnabled = false;  buttonLogin.IsEnabled = false;  MessageBox.Show("Блокировка 30 секунд");  timer.Start(); }  else MessageBox.Show($"Число оставшихся попыток - {tries} ");  }  }  else  {  MessageBox.Show("Заполните поля");  }  } |

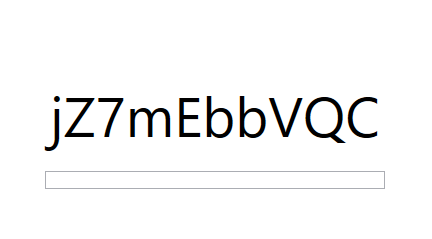


Рисунок 4 - окно каптчи

В окне каптчи генерируется случайная последовательность из 10 символов, пользователь вводит каптчу, при неудачном вводе создается новая последовательность символов.

Листинг 2 - листинг метода генерации каптчи

|  |
| --- |
| string FillCapctha()  {  string combination = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";  StringBuilder captcha = new StringBuilder();  for (int i = 0; i < 10; i++)  {  captcha.Append(combination[random.Next(combination.Length)]);  }  return captcha.ToString();  } |

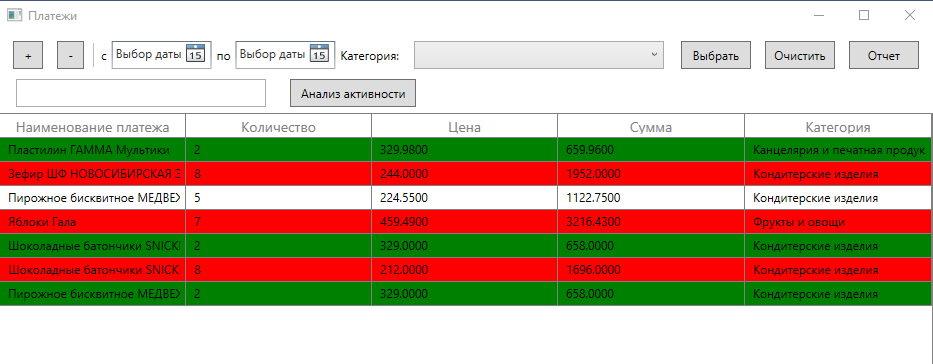


Рисунок 5 - Окно Платежи

На окне с платежами отображаются платежи авторизованного пользователя, имеется возможность добавления, изменения и удаления платежа. Также на форме реализован поиск и фильтрация платежей по дате и категории.

При нажатии на кнопку Отчет, формируется документ с платежами в формате pdf.

При нажатии на кнопку Анализ активности, открывается окно с отображением информации в графическом виде (Окно Анализ активности).

Отображается топ 3 самых дорогих (Красный цвет) и самых дешевых (Зеленый цвет) платежей по сумме.

Листинг 3 - листинг обработчика нажатия кнопки добавления/изменения платежа

|  |
| --- |
| private void buttonAdd\_Click(object sender, RoutedEventArgs e)  {  Window addPaymentWindow;  payments payment = dataGridPayments.SelectedItem as payments;  addPaymentWindow = new AddPaymentWindow(currentUser, payment);  bool result = (bool)addPaymentWindow.ShowDialog();  if (!result)  {  if (payment != null) editPaymentsCount++;  else addPaymentsCount++;  ClearData();  }  } |

Листинг 4 - листинг метода удаления платежей

|  |
| --- |
| private void removeItems(List<payments> paymentsForRemoving)  {  try  {  Instances.db.payments.RemoveRange(paymentsForRemoving);  Instances.db.SaveChanges();  MessageBox.Show("Данные удалены");  dataGridPayments.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == currentUser.PK\_users\_id).ToList();  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  } |

Листинг 5 - листинг метода поиска и фильтрации платежей

|  |
| --- |
| void UpdatePayments()  {  paymentsOfUser = Instances.db.payments.Where(p => p.FK\_user\_id == currentUser.PK\_users\_id).ToList();  if (comboBoxCategory.SelectedIndex > -1)  {  paymentsOfUser = paymentsOfUser.Where(p => p.products.FK\_category\_id == (comboBoxCategory.SelectedItem as categories).PK\_category\_id).ToList();  }  if(datePickerFrom.SelectedDate!=null)  paymentsOfUser = paymentsOfUser.Where(p => p.date >= datePickerFrom.SelectedDate).ToList();  else if (datePickerTo.SelectedDate != null)  {  paymentsOfUser = paymentsOfUser.Where(p => p.date <= datePickerTo.SelectedDate).ToList();  }  else if (datePickerTo.SelectedDate != null && datePickerFrom.SelectedDate != null)  paymentsOfUser = paymentsOfUser.Where(p => p.date <= datePickerTo.SelectedDate && p.date >= datePickerFrom.SelectedDate).ToList();  paymentsOfUser = paymentsOfUser.Where(p => p.products.product\_name.ToLower().Contains(textBoxSearch.Text.ToLower())).ToList();  dataGridPayments.ItemsSource = paymentsOfUser;  } |

Листинг 6 - листинг обработчика нажатия кнопки создания отчета

|  |  |
| --- | --- |
| private void buttonReport\_Click(object sender, RoutedEventArgs e)  {  var document = dataGridPayments.ExportToPdf();  document.Save($"Report { DateTime.Now.ToString("d")}.pdf");  MessageBox.Show("Отчет сохранен");  }   |  | | --- | | public object Convert(object value, Type targetType, object parameter, CultureInfo culture)  {  payments payment = (payments)value;  var paymentsOfUser = Instances.db.payments.Where(p => p.FK\_user\_id == payment.FK\_user\_id).ToList();  var top3 = paymentsOfUser.OrderByDescending(p => p.sum).ToList();  var top3Bottom = paymentsOfUser.OrderBy(p => p.sum).ToList();  top3Bottom = top3Bottom.GetRange(0, 3).ToList();  top3 = top3.GetRange(0, 3).ToList();  if(top3Bottom.LastOrDefault().sum >= payment.sum)  return "Green";  if (top3.LastOrDefault().sum <= payment.sum)  return "Red";  return "White";  } | |

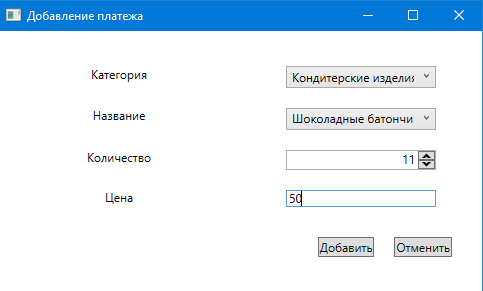


Рисунок 6 - Окно добавления/изменения платежа

На форме добавления платежа реализован выбор категории и названия продукта в выпадающем списке, присутствуют 2 текстовых поля для ввода количества и цены. При выборе конкретного платежа имеется возможность его редактирования.

При нажатии на кнопку Добавить, платеж добавляется/изменяется в системе.

При нажатии на кнопку Отменить, окно закрывается.

Листинг 7 - листинг обработчика нажатия кнопки Добавить

|  |
| --- |
| private void buttonAdd\_Click(object sender, RoutedEventArgs e)  {  if (comboBoxCategory.SelectedIndex > -1 && comboBoxProduct.SelectedIndex > -1 &&  numericUpDownCount.Value > 0 && double.TryParse(textBoxPrice.Text, out double Value))  {  int id = Instances.db.payments.ToList().Count;  string paymentId = $"{(comboBoxProduct.SelectedItem as products).categories.category\_name.ToString().Substring(0, 1)}-{id + 1}-{DateTime.Now.ToString("d")}";  try  {  if(pkId==0)  {  var paymentNew = new payments()  {  payment\_key = paymentId,  count = (int)numericUpDownCount.Value,  price = (decimal)Convert.ToDouble(textBoxPrice.Text),  sum = (decimal)(numericUpDownCount.Value \* Convert.ToDouble(textBoxPrice.Text)),  FK\_product\_id = (comboBoxProduct.SelectedItem as products).PK\_product\_id,  FK\_user\_id = user.PK\_users\_id,  date = DateTime.Now,  purpose = (comboBoxCategory.SelectedItem as categories).category\_name  };  Instances.db.payments.Add(paymentNew);  }  else  {  var payment = Instances.db.payments.Where(p => p.PK\_payment\_id == pkId).FirstOrDefault();  payment.count = (int)numericUpDownCount.Value;  payment.price = (decimal)Convert.ToDouble(textBoxPrice.Text);  payment.sum = (decimal)(numericUpDownCount.Value \* Convert.ToDouble(textBoxPrice.Text));  payment.FK\_product\_id = (comboBoxProduct.SelectedItem as products).PK\_product\_id;  payment.FK\_user\_id = user.PK\_users\_id;  payment.date = DateTime.Now;  payment.purpose = (comboBoxCategory.SelectedItem as categories).category\_name;  Instances.db.SaveChanges();  }  Instances.db.SaveChanges();  MessageBox.Show("Успех");  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  else MessageBox.Show("Проверьте введенные данные!");  } |

**Описание отчетов**

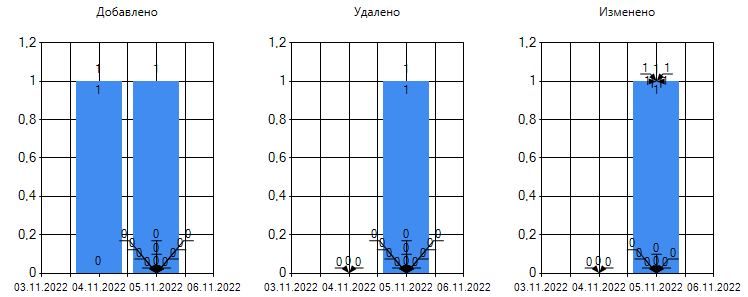
**

Рисунок 7 - Графический анализ активности по дням.

|  |
| --- |
| Листинг 8 – фрагмент кода создания графиков  chartAnalysisAdded.ChartAreas.Add(new ChartArea("Main"));  chartAnalysisChanged.ChartAreas.Add(new ChartArea("Main"));  chartAnalysisDeleted.ChartAreas.Add(new ChartArea("Main"));  var seriesAdded = new Series("analysisAdded")  {  IsValueShownAsLabel = true  };  var seriesChanged= new Series("analysisChanged")  {  IsValueShownAsLabel = true  };  var seriesDeleted = new Series("analysisDeleted")  {  IsValueShownAsLabel = true  };  chartAnalysisAdded.Series.Add(seriesAdded);  chartAnalysisDeleted.Series.Add(seriesDeleted);  chartAnalysisChanged.Series.Add(seriesChanged);  var analyzes = Instances.db.analyzes.Where(p => p.FK\_user\_id == currentUser.PK\_users\_id).ToList();  for(int i = 0; i < analyzes.Count; i++)  {  seriesAdded.Points.AddXY(analyzes[i].date, analyzes[i].added\_count);  seriesDeleted.Points.AddXY(analyzes[i].date, analyzes[i].deleted\_count);  seriesChanged.Points.AddXY(analyzes[i].date, analyzes[i].changed\_count);  } |

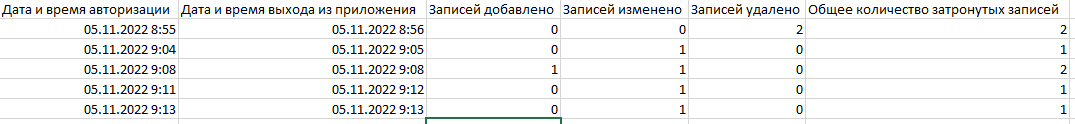
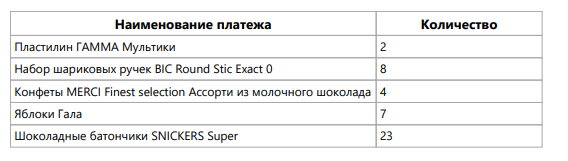


Рисунок 8. Табличный вид активности по дням

Листинг 9 - листинг метода создания табличного отчета

|  |
| --- |
| void MakeReport(string path, bool exists)  {  List<String> stringsInfo = new List<string>() { "Дата и время авторизации", "Дата и время выхода из приложения", "Записей добавлено", "Записей изменено", "Записей удалено", "Общее количество затронутых записей" };  List<String> listResults = new List<string>() { $"{dateAuthorization}", $"{DateTime.Now}", $"{addPaymentsCount}", $"{editPaymentsCount}", $"{deletePaymentsCount}", $"{addPaymentsCount + editPaymentsCount + deletePaymentsCount}" };  using (StreamWriter streamWriter = new StreamWriter(path, true, Encoding.Default, 10))  { using (CsvWriter csvWriter = new CsvWriter(streamWriter, CultureInfo.GetCultureInfo("ru-RU")))  { if (!exists) csvWriter.WriteField(stringsInfo);  csvWriter.NextRecord();  csvWriter.WriteField(listResults);  } } } |



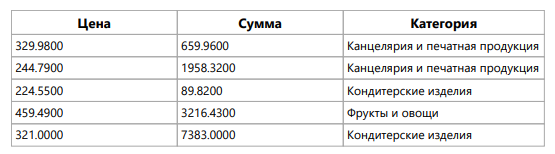


Рисунок 9-10 - Отчет по платежам

**Заключение**

В течение практики УП.11.02 Разработка приложения была проделана следующая работа: по выданной предметной области была построена логическая и физическая модель данных, созданная база данных была интегрирована в среду Microsoft Visual Studio с помощью технологии Entity Framework, в дальнейшем было разработано программное решение для учета платежей физических лиц с возможностью манипулирования данными и вывода их в отчеты в табличных и графических видах. В результате я приобрел навыки работы, необходимые для разработки готовых трехуровневых приложений.

# Используемые источники информации

1. Stackoverflow [Электронный ресурс]. – URL: https://stackoverflow.com;
2. CodeRoad [Электронный ресурс]. – URL: <https://coderoad.ru>;
3. Metanit [Электронный ресурс]. – URL: <https://metanit.com>;
4. CyberForum [Электронный ресурс]. – URL: <https://www.cyberforum.ru/>;
5. Habr [Электронный ресурс]. – URL: https://habr.com/ru/all/
6. Microsoft docs [Электронный ресурс]. – URL: https://learn.microsoft.com/ru-ru/
7. Code project [Электронный ресурс]. – URL: https://www.codeproject.com/
8. Professor Web [Электронный ресурс]. – URL: <https://professorweb.ru/>
9. MSDN [Электронный ресурс]. – URL: <https://social.msdn.microsoft.com/>
10. C-sharpcorner [Электронный ресурс]. – URL: https://www.c-sharpcorner.com/