Частное учреждение образование

«Колледж бизнеса и права»

ОТЧЕТ по ПРАКТИКЕ

по РАЗРАБОТКЕ и сопровождению программного обеспечения

ОП Т.794005

Выполнил Г.С.Выставкин

Руководитель практики В.Ю.Михалевич

2020

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.794005

Разраб.

*Выставкин Г.С.*

Провер.

Михалевич В.Ю

Т. Контр.

Н. Контр.

Утверд.

Отчет по практике

по разработке и сопровождению программного обеспечения

Лит.

Листов

49

КБП

1 Программа практики 3

2 Объектно-ориентированный анализ и проектирование системы 4

2.1 Сущность задачи 4

2.2 Проектирование модели 6

3 Вычислительная система 12

3.1 Требования к аппаратным и операционным ресурсам 12

3.2 Инструменты разработки 12

4 Проектирование задачи 14

4.1 Требования к приложению 14

4.2 Концептуальный прототип 15

4.3 Организация данных 17

4.4 Функции и элементы управления 18

4.5 Проектирование справочной системы приложения 20

5 Описание программного средства 22

5.1 Общие сведения 22

5.2 Функциональное назначение 22

5.3 Входные и выходные данные 22

6 Программирование на языке С# в среде VisualStudio 24

6.1 Технические требования 24

6.2 Функциональное тестирование 24

7 Применение 31

7.1 Назначение программы 31

7.2 Условия применения 31

7.3 Справочная система 31

Заключение 34

Список информационных источников 35

Приложение А Код программы 36

Приложение Б Формы выходных документов 49

## 1 Программа практики

Целями практики по разработке и сопровождению программного обеспечения являются:

– закрепление знаний, связанных с технологией обработки информации;

– приобретение навыков разработки программ с использованием современных средств обработки экономической и деловой информации;

– выработка умений применять средства стандартных библиотек в практических задачах.

Задачами практики по разработке и сопровождению ПО являются:

– углубленное изучение принципов организации программного обеспечения и технологии его проектирования;

– владение методами надежного программирования;

– умение разрабатывать программы в соответствии с промышленными требованиями, обеспечивая высокий уровень качества программного обеспечения и экономической эффективности;

– изучение способов определения экономической себестоимости и эффективности разработки программного обеспечения;

– оформление комплекта документации на созданное программное обеспечение.

Таблица 1 – Календарный график работы

|  |  |
| --- | --- |
| Период | Наименование и содержание работ |
| 09.02.20 | Изучение требований к рабочей дисциплине и технике безопасности.  Изучение нормативных документов. |
| 11.02.20-  17.02.20 | Ознакомление с различным программным обеспечением, используемым для разработки программ |
| 18.02.20-  24.02.20 | Получение индивидуальных заданий по созданию программ для обработки экономической и деловой информации. Обсуждение тем индивидуальных заданий. |
| 25.02.20-  03.03.20 | Изучение общих требований к программному обеспечению обработки экономической и деловой информации. Требования к интерфейсу пользователя, выходным документам (печатные, экранные формы, форматы результирующих файлов с результатами расчетов). |
| 04.03.20-  24.03.20 | Анализ предметной области индивидуального задания. Выбор и обоснование языка и среды программирования. Описание функциональных возможностей задачи. |
| 25.03.20-  07.04.20 | Технология создания программного обеспечения. Правила построения диаграмм UML. Проектирование ПО при помощи CASE-средства Rational Rose |
| 08.04.20-  21.04.20 | Конструирование алгоритма и его реализация в выбранной визуальной среде программирования. Разработка интерфейса .Проектирование классов, определение свойств и методов, определение связей между классами. |
| 22.04.20-  05.05.20 | Создание информационной базы задачи. Тестирование и отладка программы с использованием подготовленных тестовых примеров. Проектирование модулей, позволяющих осуществить защиту программного продукта от несанкционированного доступа.. |
| 06.05.20-  26.05.20 | Критерии оценки качества программного продукта. Изучение нормативных документов, регламентирующих оформление технического проекта. Создание документации по эксплуатации и сопровождению разработанного ПС. Проведение испытания ПС в нормальных и критических условиях.. |
| 27.05.20-  02.06.20 | Анализ возможности модернизации проекта с учетом альтернативных решений. Оформление аннотации на программное средство. |
| 03.06.20-  16.06.20 | Оформление отчета по практике. Подготовка листингов программ с дополнением комментариев. Распечатка экранных форм. |
| 17.06.20-  26.06.20 | Демонстрация программ по индивидуальным заданиям.  Защита отчета по практике. |

2 Объектно-ориентированный анализ и проектирование системы

2.1 Сущность задачи

Предметной областью курсового проекта является учет доходов и расходов. Данную тему можно отнести к одному из главных принципов финансовой грамотности.

Финансовая грамотность **–** набор компетенций человека, которые образуют основу для разумного принятия финансовых решений. Считается, что развитие финансовой грамотности дает возможность поддерживать и улучшать финансовое благополучие.

Финансовая грамотность включает в себя четыре ключевые области:

**–** деньги и сделки;

**–** планирование и управление финансами;

**–** риск и вознаграждения;

**–** финансовый ландшафт.

Уровень финансовой грамотности в целом – недостаточен, а в рамках отдельных социальных групп (женщины, пожилые люди и люди с низким уровнем образования) эта проблема стоит особенно остро. Дополнительным фактором риска является общее усложнение финансовых отношений, увеличение ответственности, перекладываемой государством на граждан по вопросам выбора пенсионного обеспечения, способов получения заемных средств и т. п. В рамках своей обыденной деятельности люди экспериментальным путем вырабатывают оптимальные алгоритмы действий. Но важные события (вроде взятие ипотеки, получение образования, формирование долгосрочных инвестиций) происходят редко и поэтому собственный опыт человека часто отсутствует. Именно здесь проявляется важность специфических финансовых знаний, получаемых извне. Финансовая грамотность теоретически является одним из путей улучшения экономической ситуации. Однако по мнению исследователей: строгое регулирование, обеспечение стимулов для улучшения архитектуры выбора, упрощение раскрытия информации о ценах, условиях или характеристиках продуктов, обеспечение стимулов для принятия мер **–** дают тот же эффект при меньших расходах.

Доход **–** денежные средства или материальные ценности, полученные государством, физическим или юридическим лицом в результате какой-либо деятельности за определённый период времени.

К доходам относятся:

**–** заработная плата;

**–** пенсия;

**–** стипендия;

**–** пособие;

**–** доход от продажи товаров;

**–** произведённых в собственном хозяйстве;

**–** денежные поступления в виде платы за оказанные услуги;

**–** авторские гонорары;

**–** доход от продажи личного имущества;

**–** сдачи его в аренду.

Расход средств – это направление денежных ресурсов на определенные, как правило, ранее запланированные цели.

К расходам относятся те траты, которые мы осуществляем. Расходы бывают обязательными (необходимыми) и необязательными. К обязательным (необходимым) относятся те расходы, которые удовлетворяют насущные потребности – это расходы за коммунальные услуги, расходы на питание, одежду, телефон, транспорт. К необязательным относят расходы, связанные с удовлетворением своих желаний, – то, что вы хотите купить, но от приобретения чего вы можете отказаться в любой момент времени. Здесь надо заметить, что необязательные расходы могут быть хорошим стимулом к сбережению.

Учет доходов и расходов – один из главных принципов финансовой грамотности, который подразумевает перед собой контроль собственных доходов и расходов для их контроля, оптимизации и т.д. Есть несколько важных причин, зачем вести учет и планировать бюджет:

Это делается чтобы:

1. Знать свои реальные доходы и расходы в течение определенного периода (месяца или года). Ведение бюджета поможет увидеть, сколько вы реально зарабатываете, и на что ежемесячно расходуются средства. В течение нескольких месяцев можно отслеживать структуру своих доходов, выявить преобладающие в ней поступления (заработная плата, доход от предпринимательства, хобби или то, что вы взяли в долг у родителей). Вместе с этим, вы получите полную информацию о своих тратах. Проводимый анализ позволит найти пути их снижения.

2. Находить «финансовые бреши», через которые утекают ваши деньги. Это те самые импульсивные и необдуманные покупки и вложения, которые не принесли никакой выгоды. Теперь вы сможете легко от них отказаться, так как наглядно увидите, сколько денег тратится впустую ежемесячно. Бюджет поможет более эффективно расходовать заработанное и сокращать траты.

3. Начать планировать свой бюджет. Это важный шаг на пути к управлению своими расходами. Планирование поможет использовать деньги осознанно, обходиться без дорогостоящих кредитов и обдумывать траты. Если покупка запланирована, то до ее приобретения будет определенное время, в течение которого можно отложить необходимую сумму. Для тех, кто имеет постоянный доход, составление четкого плана расходов поможет открыть новые возможности для вложения свободных денежных средств, расплатиться с долгами и кредитами и научиться откладывать деньги на поставленные цели.

4. Оптимизировать траты и найти источники дополнительной выгоды. Из плана доходов и расходов за прошедшие месяцы станет ясно, какие траты в семье являются основными, а какие второстепенными. Какие платежи нужно делать обязательно (например, за ЖКХ, за обучение и кружки детей, налоги), а от каких расходов можно отказаться (например, покупка «вкусняшек» по дороге с работы или кофе по пути на работу – любопытно, что именно отказ от утреннего кофе во многих статьях ставится чуть ли не на первое место в оптимизации своих трат).

В 2016 году ОЭСР опубликовала результаты международного сравнительного исследования финансовой грамотности в 30 странах. По результатам исследования средняя оценка финансовой грамотности по всем странам – 13,2 баллов из максимальных 21. В странах-участницах ОЭСР средний показатель **–** 13,7 баллов. Средняя оценка по странам G20 **–** 12,7 балла.

Баланс счета – это фактический остаток денежных средств на счёте по состоянию на конкретную дату.

Счёт **–** счёт, открываемый юридическим или физическим лицом для их участия в безналичном денежном обороте и аккумулировании на счёте безналичных денежных средств для целевого использования.

Остаток средств **–** сумма, числящаяся на счете юридического или физического лица с учетом расходов со счета и поступлений на счет.

Для учёта доходов и расходов используется формула 1:

(1)

где, доходы – сумма полученных средств (зарплата, пенсия и т.д.)

расходы – сумма израсходованных средств (ЖКХ, квартплата и т.д.)

баланс – окончательное значение после расчёта разницы между доходами и расходами

Основные функции программы:

**–** добавление доходов и расходов

**–** планирование бюджета

**–** просмотр всех балансов

**–** счетов и остатков средств.

Выводы о функциях программы были сделаны из предметной области.

Программами-аналогами являются CoinKeeper, Monefy, PingFin, Дзен-мани: учет расходов и др. В своей сути данные программы имеют одинаковый функционал, но в каждой также имеется своя «изюминка», свой удобный интерфейс, различные функции, которые могут быть удобнее пользователю и др.

Создаваемое программное средство имеет следующие отличия: мультивалютность, планирование бюджета, наглядные графики, диаграммы, которые покажут, куда были израсходованы средства.

График, в данном случае будет отражать подневную сумму затрат, например, за 25 января – 25 рублей. Планируется реализовать график так: график будет похож на линию координат. На одной оси будет отображаться сумма затраты, а на другой – день, в который были затрачены средства. Также ниже будет отображение в процентах, сколько затрачено и на какую категорию. С примерным графиком можно ознакомиться на рисунке 1.

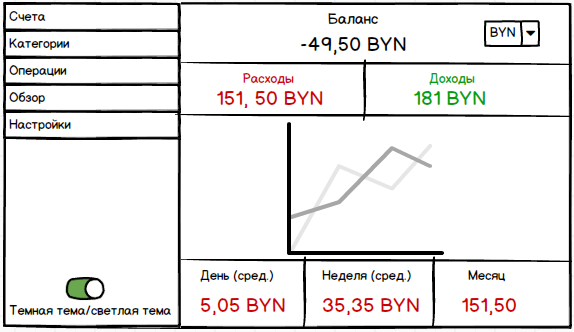


Рисунок 1 – Макет приложения в разделе «Обзор»

2.2 Проектирование модели

ER-модель **–** модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь».

ERD-диаграмма по Баркеру располагается на рисунке 1. ERD-диаграмма по Чену располагается на рисунке 2.

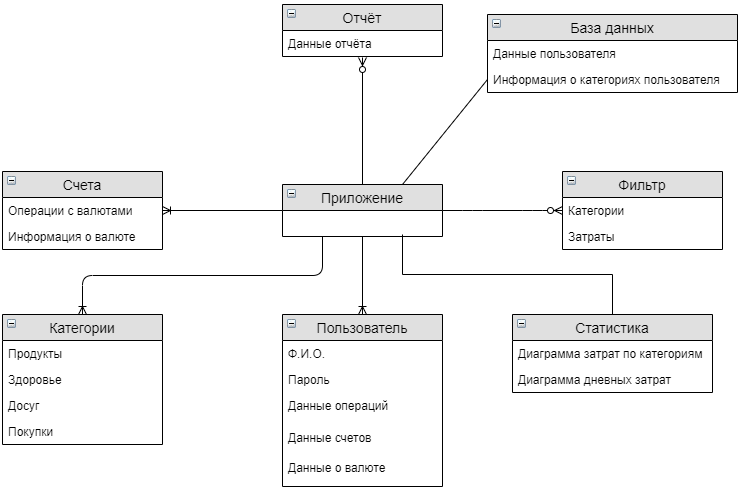


Рисунок 2 – ERD-Диаграмма по Баркеру

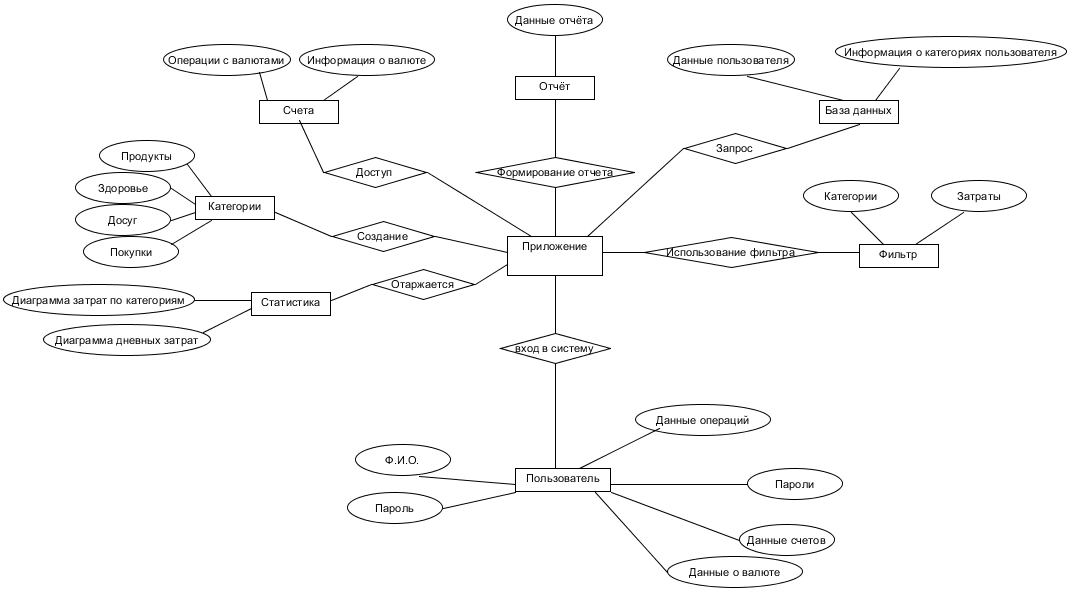


Рисунок 3 – ERD-Диаграмма по Чену

Сущность «Пользователь» состоит из атрибутов:

**–** Ф.И.О.

**–** пароль

**–** данные операций

**–** данные о валюте

**–** данные счетов

Сущность «Категории» состоит из атрибутов:

**–** продукты

**–** здоровье

**–** досуг

**–** покупки

Сущность «Статистика» состоит из атрибутов:

**–** диаграмма затрат по категориям

**–** диаграмма дневных затрат

Сущность «Фильтр» состоит из атрибутов:

**–** категории

**–** затраты

Сущность «Счета» состоит из атрибутов:

**–** операции с валютами

**–** информация о валюте

Сущность «База данных» состоит из атрибутов:

**–** данные пользователя

**–** информация о категориях пользователя

Сущность «Отчёт» состоит из атрибутов:

**–** данные отчёта

Сущность «Приложение» состоит из атрибутов:

**–** приложение

Диаграмма вариантов использования **–** диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент **–** возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Основное назначение диаграммы **–** описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Отношение расширения **–** определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров; отношение является направленным и указывает, что применительно к отдельным вариантам использования выполнены конкретные условия и что один из вариантов может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта; отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой, направленной от варианта использования, который является расширением. Сама линия помечается ключевым словом «extend» («расширяет»).

Отношение включения **–** указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данное отношение является направленным бинарным отношением, при котором, когда экземпляр первого варианта использования в процессе своего выполнения достигает точки включения в последовательность поведения экземпляра второго варианта использования, он выполняет последовательность действий, определяющий поведение второго варианта, после чего продолжает выполнение действий своего поведения. Один вариант может быть включен в несколько других вариантов, а также включать в себя другие варианты.

На данной диаграмме имеется один «актёр» **–** это пользователь. Он имеет такие варианты использования как: просмотр категорий, регистрация, авторизация, просмотр счетов. В основном при создании диаграммы рассматривались такие связи как «extend», т.е. при выборе и выполнении одного определенного варианта, у пользователя расширяется диапазон выбора различных функций (добавляются дополнительные поведения).

Диаграмма вариантов использования располагается на листе 1 в графической части.

Диаграмма классов **–** структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, типов и т. п.) Она содержит в себе также некоторые элементы поведения, однако их динамика должна быть отражена на диаграммах других видов (диаграммах коммуникации, диаграммах состояний). Для удобства восприятия диаграмму классов можно также дополнить представлением пакетов, включая вложенные.

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности. Отношение композиции служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого. Подробное описание классов приложения находится в таблице 2.

Таблица 2. Описание классов приложения

|  |  |  |
| --- | --- | --- |
| Название класса | Поля класса | Функции класса |
| Пользователь | Name, Surname, Login, Password | Register (Name, Surname, Login, Password) |
| Приложение |  | Login (Login, Password) |
| Счёт | Balance, Currency, Type | Operations(), Information(), Editing(), Remove(), Adding(), Sort() |
| Категории | Products, Health, Leisure, Purchases | Editing(), Remove(), Adding(), Sort() |
| Вывод информации |  | ShowInfo(), ShowStatistics() |

Классы «Приложение», «Счёт» и «Категории» соединены между собой отношением композиции. «Пользователь» и «Приложение» **–** отношением агрегации. Обусловлено это тем, что отношение композиции связывает классы, которые обязательны и являются основными компонентами программы. В то время как те классы, которые связаны отношением агрегации, наоборот, являются вторыми по приоритету (необязательны). Диаграмма классов располагается на листе 2 в графической части.

Диаграмма последовательности **–** диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов). Используется в языке UML.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

На данной диаграмме показана последовательность действий при регистрации или авторизации в программу, а также редактирование категории. Также показан специальный блок действий, который показывает альтернативные варианты поведения в программе. В альтернативном блоке показано, как взаимодействует между собой приложение, база данных и функции приложения. Таким образом можно просмотреть, как поведет себя программа, если пользователь введёт неверные данные при регистрации или же при неправильном редактировании своей категории.

Диаграмма последовательности располагается на листе 3 в графической части.

Диаграмма деятельности **–** UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов **–** вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений. В диаграмме присутствует начало и конец какого-либо действия. Отмечается начало действия – закрашенным черным кружком и на границе кружка – черная линия. Конец действия – чуть более уменьшенный черный кружок и с небольшим отступом на границах – красная линия.

На данной диаграмме показан детальный процесс создания счёта в программе. От начала – запуска приложения и до конца – сохранения изменений. Диаграмма деятельности располагается на листе 4 в графической части.

Диаграмма компонентов –элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты.

С помощью диаграммы компонентов представляются инкапсулированные классы вместе с их интерфейсными оболочками, портами и внутренними структурами (которые тоже могут состоять из компонентов и коннекторов).

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывает связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

На данной диаграмме показан состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код, а также установить зависимости между ними. На диаграмме представлена папка с файлами программного средства, папка включает такие файлы:

– «AddCardMoney.cs» – файл добавления доходов на карту;

– «AddCash.cs» – файл добавления доходов наличными;

– «AddCategory.cs» – файл добавления расходов;

– «Database.cs» – файл подключения к БД;

– «HelpSystem.cs» – файл справочной системы;

– «MainWindow.cs» – файл главного окна;

– «MoneyDesktop.mdf» - файл базы данных;

– «MoneyDesktop\_log.ldf» - файл базы данных;

Диаграмма компонентов располагается на листе 5 в графической части.

3 Вычислительная система

3.1 Требования к аппаратным и операционным ресурсам

Данное программное средство не должно быть требовательно к ресурсам персонального компьютера. Приведенных ниже системных требований должно быть достаточно для использования:

**–** операционная система Windows 7 64-bit и выше;

**–** процессор Intel Core i3-2365M CPU @ 1.40GHz, AMD Ryzen 3 2200G

**–** ОЗУ от 512 МБ;

**–** видеокарта IntelHDGraphics 3000, AMD Vega 8;

**–** места на HDD 100 МБ;

**–** мышь, клавиатура, touchpad.

3.2 Инструменты разработки

Инструментами разработки являются:

**–** операционная система Windows 10;

**–** язык программирования C#;

**–** система для построения приложений WPF;

**–** средство построения диаграмм Draw.io;

**–** среда разработки Visual Studio Professional 2019;

**–** язык структурированных запросов SQL SMS;

**–** текстовый редактор Microsoft Word 2016

Операционная система Windows 10 является одной из самых надежных операционных систем Windows. Она имеет множество удобных приложений и функций, которых не было в предшественниках. Windows 10 занимает первое место в мире среди операционных систем, используемых для доступа к сети Интернет, опередив в апреле 2017 года предыдущего лидера **–** Windows 7 [7].

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Данный язык использует объектно-ориентированный подход к программированию во всем. Это означает, что нужно будет описывать абстрактные конструкции на основе предметной области, а потом реализовывать между ними взаимодействие. Также в языке присутствует обилие синтаксического «сахара», который делает тяжелую жизнь капельку слаще. Вместо того, чтобы писать кучу строк кода, можно просто использовать готовую конструкцию, а компилятор сделает за тебя всю «грязную» работу. Но некоторые такие конструкции являются не самыми оптимальными с точки зрения производительности. Но все это перекрывается за счет удобочитаемости кода и высокой скоростью разработки [3].

Windows Presentation Foundation (WPF) **–** это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Также выбор был сделан исходя из следующих приемуществ:

**–** независимость от разрешающей способности, аппаратное ускорение, поддержку мультисенсорного ввода.

**–** стили, триггеры, шаблоны, анимацию, аудио и видео, команды, мощную систему привязки, маршрутизируемые события.

**–** продвинутые возможности для рисования.

Кроме того, идеи, принципы и философия, которые положены в основу данной технологии, мне очень удобны, импонируют и симпатичны.

Draw.io **–** инструмент, который позволяет создавать блок-схемы, сетевые диаграммы, интеллект-карты, отношения сущностей, программные блоки, UML, макеты и т. д. Богатая функциональность Draw.io позволяет пользователям отслеживать и восстанавливать изменения, импортировать и экспортировать в PDF, PNG, XML, VSDX, HTML, а также автоматически публиковать и делиться работами.

С помощью сервиса можно создавать:

**–** графики;

**–** блок-схемы;

**–** диаграммы;

**–** ментальные карты;

**–** макеты.

Особенности:

**–** большой набор шаблонов;

**–** быстрое создание блоков

**–** интуитивно понятный интерфейс;

**–** экспорт в форматы JPG, PNG, SVG, VDSX;

**–** совместная работа;

**–** русскоязычный интерфейс;

**–** возможность скачать в виде программного средства для компьютера.

Microsoft Visual Studio **–** линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии WPF, а также веб-сайты, веб-приложения, веб-службы. Visual Studio включает в себя редактор исходного кода с возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных [8].

SQL **–** декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL считается языком программирования, в общем случае не является тьюринг-полным, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений. При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов [6].

4 Проектирование задачи

4.1 Требования к приложению

SADT – методология структурного анализа и проектирования, интегрирующая процесс моделирования, управление конфигурацией проекта, использование дополнительных языковых средств и руководство проектом со своим графическим языком. Процесс моделирования может быть разделен на несколько этапов: опрос экспертов, создание диаграмм и моделей, распространение документации, оценка адекватности моделей и принятие их для дальнейшего использования. Этот процесс хорошо отлажен, потому что при разработке проекта специалисты выполняют конкретные обязанности, а библиотекарь обеспечивает своевременный обмен информацией.

SADT – диаграмма располагается на рисунке 4.

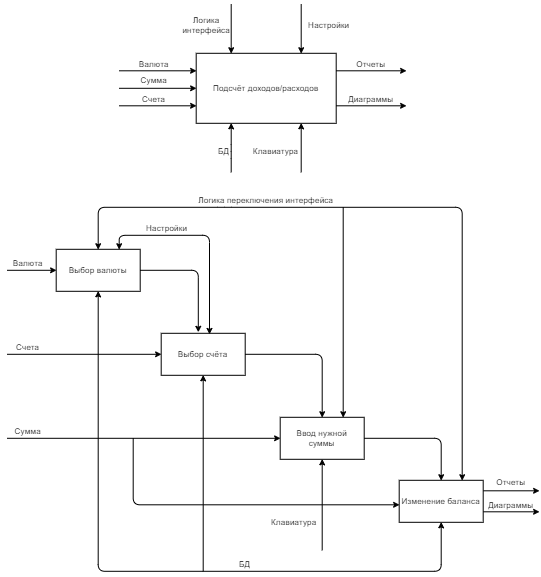


Рисунок 4 – SADT диаграмма

На данной диаграмме представлено общее представление функции (нулевой уровень) и детальное представление (первый уровень). На общем представление представлено, что входит в функцию, управление, механизмы и что мы получим на выходе данной функции. Ниже перечислены все входные и выходные данные диаграммы:

– валюта

– сумма

– счета

– логика интерфейса

– настройки

– БД

– клавиатура

– отчёты

– диаграммы

4.2 Концептуальный прототип

Концептуальный прототип – это общее видение продукта, его функций, логики взаимодействия с клиентами и другими продуктами компании.

При запуске программного средства пользователю сразу встречается простой интерфейс и начальный раздел программного средства «Счета». На рисунке 5 предоставляется начальный раздел программного средства – «Счета».

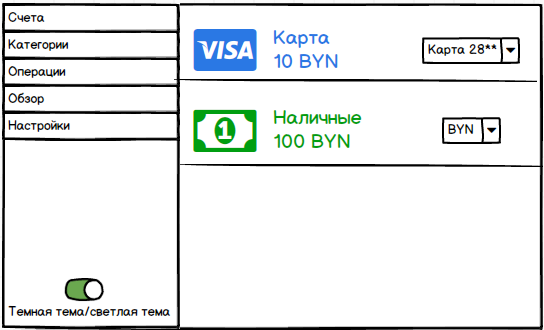


Рисунок 5 – Раздел «Счета»

Слева находися 5 разделов программного средства, в которых мы можем переключаться и просматривать их, взаимодействовать с ними. На рисунке 6 предоставляется следующий раздел программного средства – «Категории».

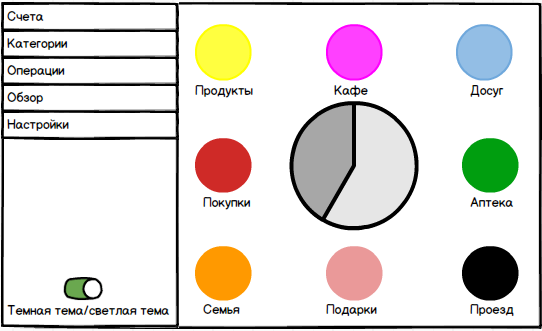


Рисунок 6 – Раздел «Категории»

При переключении на следующий раздел пользователь просмотрит раздел «Операции». Данный раздел предоставляется на рисунке 7.

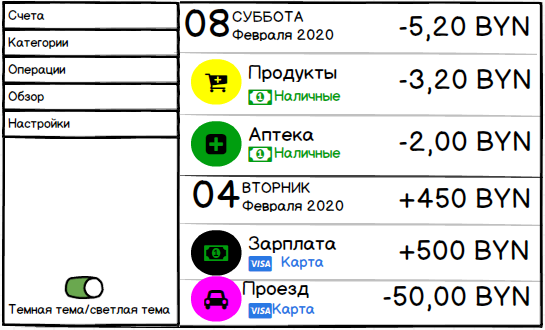


Рисунок 7 – Раздел «Операции»

При переключении на следующий раздел пользователь просмотрит раздел «Обзор». Данный раздел предоставляется на рисунке 8.

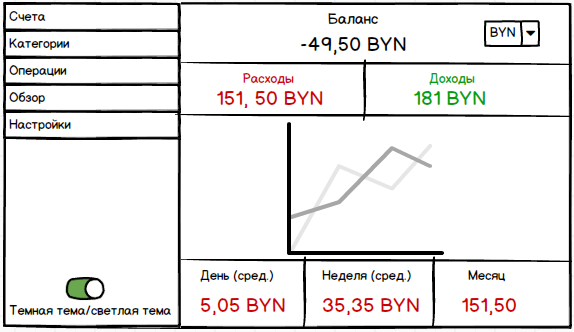


Рисунок 8 – Раздел «Обзор»

Последний раздел «Настройки» предоставлен для изменения языка, выбора языка, установки пароля и т.д. Данный раздел предоставляется на рисунке 9.

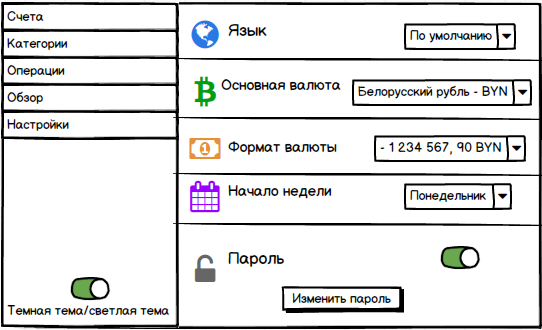


Рисунок 9 – Раздел «Настройки»

4.3 Организация данных

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы - атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

Структура базы данных разрабатываемого программного средства включает одну таблицу.

Таблица «Balance» хранит информацию о балансе пользователя, структура которой приведена в таблице 3

Таблица 3 – Структура таблицы «Balance»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | int | 4 | Идентификатор пользователя |
| Money | int | 4 | Сумма пополненного баланса пользователя |
| Card | bit | 1 | Определение использования картой или наличными |

Продолжение таблицы 3 – Структура таблицы «Balance»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| MoneySum | int | 4 | Конечная сумма пользователя |

Таблица «Expenses» хранит информацию о книгах, структура которой приведена в таблице 4.

Таблица 4 – Структура таблицы «Expenses»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Count | int | 4 | Затраченная сумма на категорию |
| Category | varchar | 255 | Название категории |
| Date | date | 8 | Дата затраченной суммы |

На рисунке 10 представлена схема базы данных.

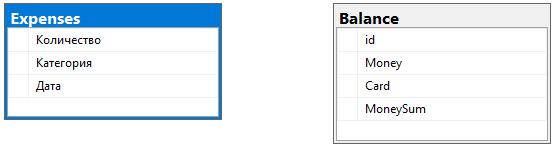


Рисунок 10 – Схема базы данных

4.4 Функции и элементы управления

Для обновления данных о сумме баланса в программном средстве необходимо перейти в «Форму категорий», на которой нужно нажать кнопку «Обновить». После нажатия кнопки «Обновить» вызывается метод Update, который обрабатывает данные обновления баланса. Данная реализация функции была реализована в средства Visual Studio Professional 2019, код которого представлен ниже. [8]

public void Update()

{

Database database = new Database();

database.OpenConnection();

SqlCommand command3 = new SqlCommand("UPDATE Balance set MoneySum = (select sum(money) from balance);", database.getConnection());

using (SqlDataReader reader = command3.ExecuteReader())

{

if (reader.Read())

{

CardSum.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command = new SqlCommand("SELECT sum(Money) from Balance where card = '1'", database.getConnection());

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

CardSum.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command1 = new SqlCommand("SELECT sum(Money) from Balance where card = '0'", database.getConnection());

using (SqlDataReader reader = command1.ExecuteReader())

{

if (reader.Read())

{

CashSum.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command2 = new SqlCommand("SELECT sum(Money) from Balance", database.getConnection());

using (SqlDataReader reader = command2.ExecuteReader())

{

if (reader.Read())

{

SumMoney.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command4 = new SqlCommand("select sum(Количество) from Expenses where Категория='Продукты'", database.getConnection());

using (SqlDataReader reader = command4.ExecuteReader())

{

if (reader.Read())

{

Label1.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command5 = new SqlCommand("select sum(Количество) from Expenses where Категория='Мобильная связь'", database.getConnection());

using (SqlDataReader reader = command5.ExecuteReader())

{

if (reader.Read())

{

Label2.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

Полный код программы располагается в приложении А.

4.5 Проектирование справочной системы приложения

Для корректной работы с приложением требуется обеспечить пользователя справочной системой, в которой будут приведены приемы работы с приложением, включающие данные о том, что произойдёт после нажатия на определенную кнопку или при выборе пункта меню.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая может напомнить принцип работы программного средства.

Структура справочной системы:

– введение – предварительные сообщения общего характера, предпосылаемые произведению, с целью ввести пользователя в курс программного средства;

– описание программного средства – основные понятия и определения для описания назначения и функционала программного средства;

– основной интерфейс программного средства – описаны главные функции программного средства;

– контекстное меню – описано как пользоваться меню программного средства;

– назначение программного средства – описано, для чего предназначено программное средство;

– о программе – описано что собой представляет программное средство.

Справочная система программного средства представляет собой отдельное окно с полным описанием основных функций программного средства. Чтобы загрузить справочную систему, нужно нажать кнопку со знаком вопроса в правой части ленты. В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Справочная система будет разработана в Visual Studio Professional 2019.

5 Описание программного средства

5.1 Общие сведения

Тестирующее программное средство на определение темперамента предназначено **для эффективного использования аппаратных и программных средств персонально компьютера и функционирования программного продукта.**

Загрузочным файлом программного средства является «WpfProject.exe».

Основными задачами программного средства являются:

– предоставление пользователю простой интерфейс;

– удобство использования приложения.

Данное программное средство было разработано на персональном компьютере со следующей конфигурацией:

– операционная система Windows 10;

– процессор Intel Core i5-3210M CPU @ 2.50GHz;

– ОЗУ 8 ГБ;

– видеокарта Intel HD Graphics 4000, NVIDIA GeForce GT 650M;

– места на HDD 1000 ГБ;

– мышь;

– клавиатура.

Программное средство создано в средстве разработки Visual Studio Professional 2019 на языке программирования C# в операционной системе Windows 10. Программное средство может работать в средах операционных систем семейства Microsoft Windows, начиная с Windows 7.

5.2 Функциональное назначение

Программное средство предназначено для учета доходов и расходов пользователя, добавления данных расходов в базу данных и взаимодействия с данными. **Данное программное средство предназначено для учета доходов и расходов, путём созданных в** программном средстве **математических функций, которые вычисляют расходы пользователя с использованием облегчённого интерфейса для удобного и быстрого пользования программой [9].**

Программное средство использует стандартные элементы управления, такие как кнопки, списки, поля ввода, что обеспечивает удобство в использовании, понятный и удобный интерфейс программного средства. **Данный программный продукт не имеет широкого назначения, поэтому сетевой поддержки не имеет.**

5.3 Входные и выходные данные

Входными данными являются данные пользователя о его расходах. Входные данные программного средства представлены в таблице 5 [10].

Таблица 5 – Входные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | Тип поля | Размер поля, байт | Описание поля |
| ID | int | 4 | Идентификатор пользователя |

Продолжение таблицы 5 – Входные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Money | int | 4 | Сумма пополненного баланса пользователя |
| Card | bit | 1 | Определение использования картой или наличными |
| MoneySum | int | 4 | Конечная сумма пользователя |
| Count | int | 4 | Затраченная сумма на категорию |
| Category | varchar | 255 | Название категории |
| Date | date | 8 | Дата затраченной суммы |

Выходными данными являются данные отчёта о затраченных средствах пользователя. Выходные данные программного средства представлены в таблице 6.

Таблица 6 – Выходные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | Тип поля | Размер поля, байт | Описание поля |
| Money | int | 4 | Сумма пополненного баланса пользователя |
| Count | int | 4 | Затраченная сумма на категорию |
| Category | varchar | 255 | Название категории |

Подробные данные о выходных данных находятся в приложении Б. Отчёт о затраченных средствах пользователя расписан на рисунке Б.1 в приложении Б.

6 Методика испытаний

6.1 Технические требования

Данная программное средство было протестировано на персональном компьютере со следующей конфигурацией:

– операционная система Windows 10;

– процессор Intel Core i5-3210M CPU @ 2.50GHz;

– ОЗУ 8 ГБ;

– видеокарта Intel HD Graphics 4000, NVIDIA GeForce GT 650M;

– места на HDD 1000 ГБ;

– мышь;

– клавиатура.

6.2 Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование всего программного средства происходит по средствам проверки всех функций, представленных в программном средстве.

Тестирование программного средства начинается с главного окна программного средства – это окно авторизации. На рисунке 11 представлено главное окно программного средства.

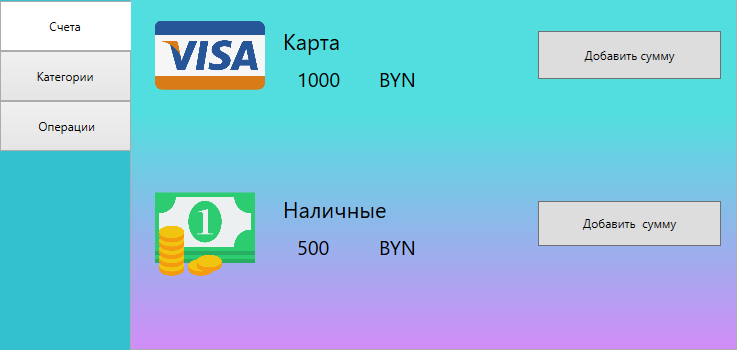


Рисунок 11 – Главное окно

Тест-кейс для добавления средств на карту приведен в таблице 7.

Таблица 7 – Тест-кейс для добавления средств на карту

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 1 | Добавление средств на карту | 1. Войти в программу; 2. Зайти в раздел «Счета»; 3. Кликнуть на кнопку «Добавить сумму»; 4. В поле «Сумма» ввести «5». | Ожидаемый:открытие главного меню с добавленными средствами. |
| Окно с заполненым полем представлено на рисунке 12.  Фактический: фактический результат представлен на рисунке 13, соответствует ожидаемому результату, все условия соблюдены. |

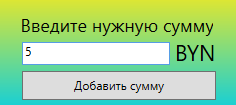


Рисунок 12 – Окно добавления суммы

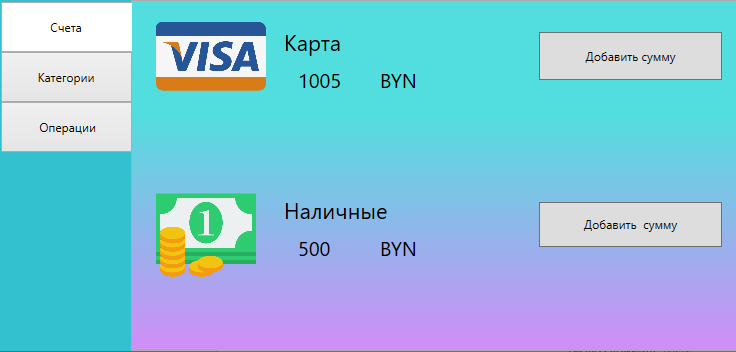


Рисунок 13 – Фактический результат

Тест-кейс для добавления средств наличными приведена в таблице 8.

Таблица 8 – Тест-кейс для добавления средств наличными

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 2 | Добавление средств наличными | 1. Войти в программу; 2. Зайти в раздел «Счета»; 3. Кликнуть на кнопку «Добавить сумму»; 4. В поле «Сумма» ввести «50». | Ожидаемый:открытие главного меню с добавленными средствами. |
| Окно с заполненым полем представлено на рисунке 14.  Фактический: фактический результат представлен на рисунке 15, соответствует ожидаемому результату, все условия соблюдены. |

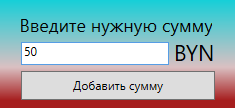


Рисунок 14 – Окно добавления суммы

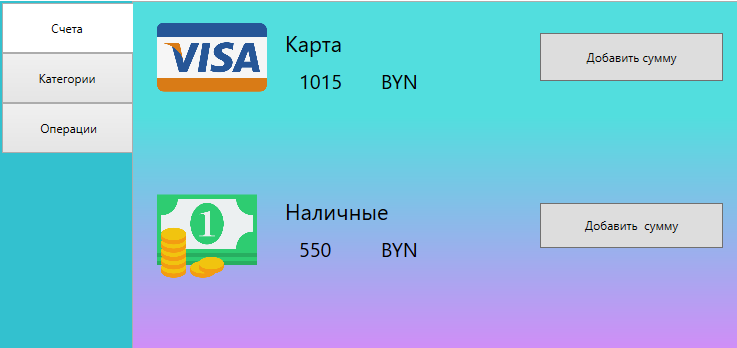


Рисунок 15 – Фактический результат

Тест-кейс для добавления расходов на категории приведена в таблице 9.

Таблица 9 – Тест-кейс для добавления расходов на категории

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 3 | Добавление расходов на категории | 1. Войти в программу; 2. Зайти в раздел «Категории»; 3. Кликнуть на категорию «Продукты»; 4. В поле «Сумма» ввести «50». 5. Кликнуть на кнопку «Добавить затрату» | Ожидаемый:открытие главного меню с добавленными расходами. |
| Окно с заполненым полем представлено на рисунке 16.  Фактический: фактический результат представлен на рисунке 17, соответствует ожидаемому результату, все условия соблюдены. |

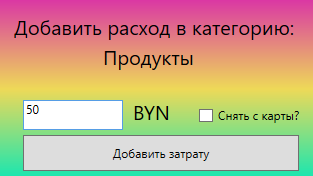


Рисунок 16 – Окно добавления расходов



Рисунок 17 – Фактический результат

Тест-кейс для просмотра таблицы затрат приведен в таблице 10.

Таблица 10 – Тест-кейс для просмотра таблицы затрат

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 4 | Просмотр таблицы затрат | 1. Войти в программу; 2. Зайти в раздел «Операции»; 3. Кликнуть на кнопку «Таблица затрат». | Ожидаемый:открытие раздела с таблицей затрат |
| Окно с главным окном программы представлено на рисунке 11.  Фактический: фактический результат представлен на рисунке 18, соответствует ожидаемому результату, все условия соблюдены. |

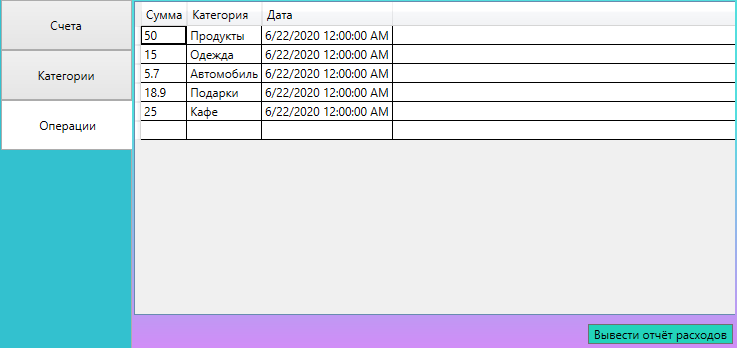


Рисунок 18 – Фактический результат

Тест-кейс для вывода отчёта о расходах приведен в таблице 11.

Таблица 11 – Тест-кейс для просмотра таблицы затрат

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 5 | Вывод отчёта о расходах | 1. Войти в программу; 2. Зайти в раздел «Операции»; 3. Кликнуть на кнопку «Вывести отчёт расходов». | Ожидаемый:вывод окна с выбором места сохранения документа |
| Окно с кнопкой представлено на картинке 18  Фактический: фактический результат представлен на рисунке 19, соответствует ожидаемому результату, все условия соблюдены. |

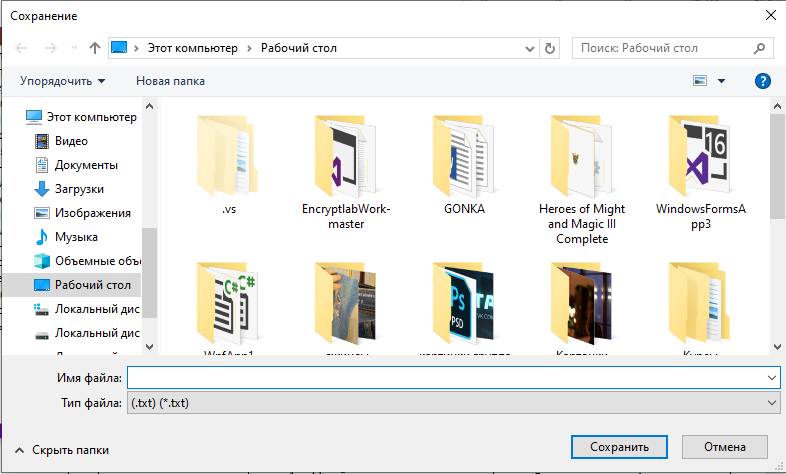


Рисунок 19 – Фактический результат

Тест-кейс просмотра справочной системы приведен в таблице 12.

Таблица 11 – Тест-кейс для просмотра таблицы затрат

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модель / Функция | Шаги воспроизведения | Результат |
| 6 | Просмотр справочной системы | 1. Войти в программу; 2. Кликнуть на кнопку «Справка». | Ожидаемый:вывод окна со справочной системой |
| Окно с кнопкой представлено на картинке 20  Фактический: фактический результат представлен на рисунке 21, соответствует ожидаемому результату, все условия соблюдены. |

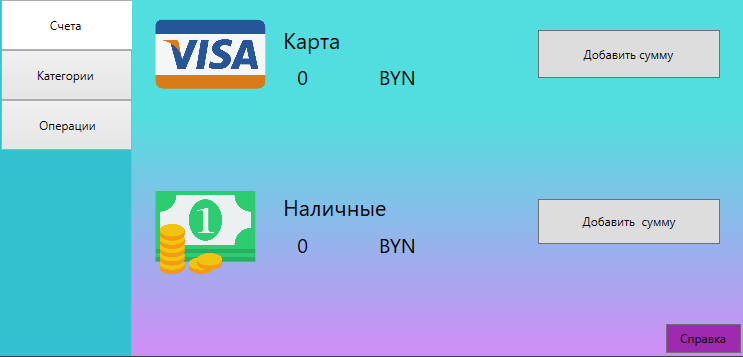


Рисунок 20 – Главное окно программного средства

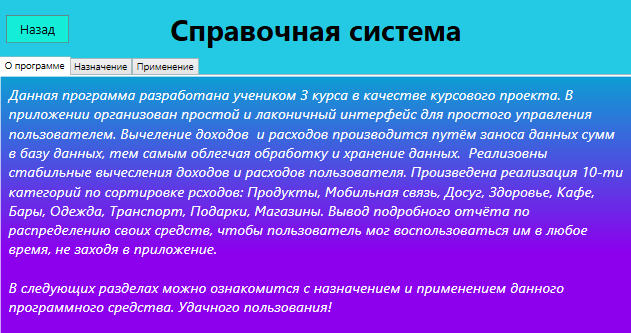


Рисунок 21 – Фактический результат

7 Применение

7.1 Назначение программы

Программное средство книжный магазин создана для контроля доходов и расходов пользователя и выбора оптимального уровня затрат. Хранение информации в базе данных позволяет значительно ускорить работу программы.

Данное программное средство решает следующие задачи:

– просмотр доходов

– просмотр расходов

– добавление доходов

– добавление расходов

– управление категориями расходов

– отчётность расходов

7.2 Условия применения

Приведенных ниже минимальных системных и программных требований будет достаточно для функционирования программного средства:

– операционная система начиная с Windows 7;

– процессор от Intel Pentium 4 – 2365M CPU @ 1.30GHz;

– ОЗУ от 500 МБ;

– видеокарта от Palit GeForce 610;

– места на HDD 100 МБ;

– мышь / touchpad;

– клавиатура;

– программная платформа Microsoft .Net Framework 4.7;

– офисный пакет приложений Microsoft Office Access Professional 2019.

7.3 Справочная система

Для корректной работы с приложением была создана для пользователя справочная система, в которой приведены приемы работы с приложением, включающие данные о том, что произойдёт после нажатия на определенную кнопку или при выборе пункта меню.

Справочная система была создана для ознакомления с программой. В ней присутствует информация, которая может напомнить принцип работы программного средства.

Структура справочной системы:

– о программе – предварительные сообщения общего характера, предпосылаемые произведению, с целью ввести пользователя в курс программного средства;

– назначение программы – описано предназначение программного средства, также ввод в основные тезисы;

– применение – описан основной сценарий использования, введенные ограничения и описание как пользоваться программой.

Справочная система программного средства представляет собой отдельное окно с полным описанием основных функций программного средства. Чтобы загрузить справочную систему, нужно нажать кнопку со знаком вопроса в правом нижнем углу программного средства. В справочной системе даны ответы на типичные вопросы, возникающие при работе с программным средством, что, несомненно, должно помочь при освоении программного средства.

На рисунке 22 представлен вводный раздел «О программе».

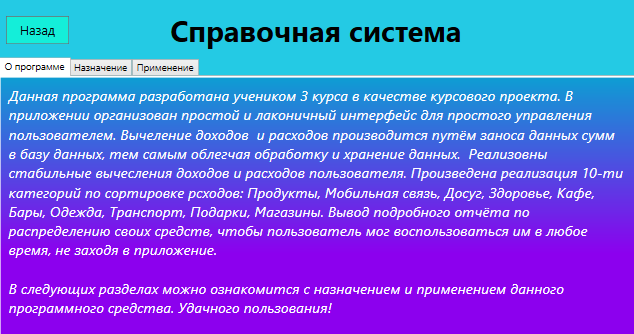


Рисунок 22 – Справочный раздел «О программе»

На рисунке 23 представлен справочный раздел «Назначение».

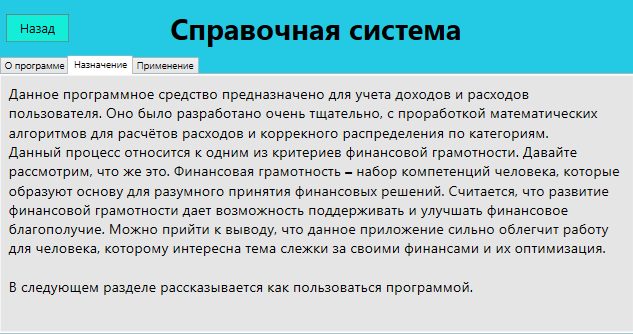


Рисунок 23 – Справочный раздел «Назначение»

На рисунке 24 представлен справочный раздел «Применение».

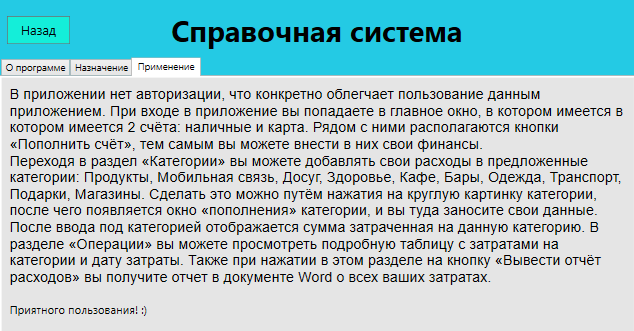


Рисунок 24 – Справочный раздел «Применение»

Заключение

В рамках учебной практики по разработке и сопровождению ПО на тему «Программное средство по учету доходов и расходов» было разработано программное средство «WpfProject.exe», позволяющее производить учет доходов и расходов, разносить затраты по категориям, выводить отчёты затрат, просматривать общий баланс и баланс счетов.

Для достижения цели в рамках учебной практики по разработке и сопровождению ПО были решены следующие задачи:

– определена вычислительная система, необходимая для создания программного средства;

– разработана физическая и логическая модель данных;

– по модели выполнено проектирование задачи;

– разработано программное средство;

– описано созданное программное средство;

– выбрана методика испытаний;

– описан процесс тестирования;

– приведены примеры области применения.

Проект был разработан в среде Visual Studio Professional 2019 на языке C#. База данных была реализована в Microsoft SQL Server Management Studio 18.

Программное средство имеет свои недостатки, такие как:

– отсуствует мультивалютность;

– отсуствует графическое отображение расходов;

– отсуствует выставление пароля для защиты от кражи информации.

А также свои преимущества:

– простой интерфейс;

– удобство использования;

– информативность.

Программное средство реализовано в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

Программное средство готово к практическому использованию и может быть дополнено и модернизировано.

Список информационных источников

1 Михалевич В.Ю. Методические указания к курсовому проектированию для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» / В.Ю. Михалевич. – Минск: КБП, 2020

2 Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2013

3 Программа и методы испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. Совет по стандартизации, метрологии и сертификации, 2000. – 16 с.

4 Руководство по программированию на C# [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : http://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/. – Дата доступа : 15.03.2020.

5 SQL Задачи и решения [Электронный ресурс]. – Sql-Tutorial, 2020. – Режим доступа : http://www.sql-tutorial.ru. – Дата доступа : 22.02.2010.

6 Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://www.microsoft.com/ru-ru/windows/features/. – Дата доступа : 18.03.2020.

7 Visual Studio 2019 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://visualstudio.microsoft.com/ru/vs/. – Дата доступа : 23.04.2020.

8 Уроки C# .NET Windows Forms Авторизация пользователя через базу данных [Электронный ресурс]. – Youtube, 2015. Режим доступа : https://www.youtube.com/watch?v=gnvxVv60xnI&t. Дата доступа: 04.02.2020.

9 Winforms Boutique Dark UI Demo Template (UI + Dataviz) [Электронный ресурс]. – Youtube, 2015. Режим доступа : https://www.youtube.com/watch?v=pnlSDH0SCHo&t. Дата доступа: 12.02.2020.

10 How to Create Quiz Application in C#.NET using SQL Server [Электронный ресурс]. – Youtube, 2015. Режим доступа : https://www.youtube.com/watch?v=\_GX7A2jIovI&t. Дата доступа: 24.03.2020.

Приложение А

(обязательное)

Текст программных модулей

//AddCardMoney.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WpfProject

{

/// <summary>

/// Логика взаимодействия для AddCardMoney.xaml

/// </summary>

public partial class AddCardMoney : Window

{

MainWindow f;

public AddCardMoney(MainWindow mw1)

{

f = mw1;

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Database database = new Database();

SqlCommand command = new SqlCommand("INSERT INTO Balance (Money,Card) VALUES (@money,1)", database.getConnection()); //подключение к бд

command.Parameters.Add("@money", SqlDbType.VarChar).Value = AddMoneyField.Text;

database.OpenConnection(); //открытие подключения

if (command.ExecuteNonQuery() == 1)

MessageBox.Show("Создано!");

else

MessageBox.Show("Не удачно! :(");

database.CloseConnection();

f.Update();

}

private void Window\_Closed(object sender, EventArgs e)

{

f.Update();

}

}

}

//AddCash.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WpfProject

{

/// <summary>

/// Логика взаимодействия для AddCash.xaml

/// </summary>

public partial class AddCash : Window

{

MainWindow f;

public AddCash(MainWindow mw)

{

f = mw;

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Database database = new Database();

SqlCommand command = new SqlCommand("INSERT INTO Balance (Money) VALUES (@money)", database.getConnection()); //ввод данных в бд

command.Parameters.Add("@money", SqlDbType.VarChar).Value = AddCashField.Text;

database.OpenConnection();

if (command.ExecuteNonQuery() == 1)

MessageBox.Show("Создано!");

else

MessageBox.Show("Не удачно! :(");

database.CloseConnection();

f.Update();

}

private void Window\_Closed(object sender, EventArgs e)

{

f.Update();

}

}

}

//AddCategory.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WpfProject

{

/// <summary>

/// Логика взаимодействия для AddCategory.xaml

/// </summary>

public partial class AddCategory : Window

{

string cat;

MainWindow f;

public AddCategory(string Category, MainWindow mw)

{

cat = Category;

f = mw;

InitializeComponent();

CategoryLabel.Content = cat;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Database database = new Database();

if (CardCheckBox.IsChecked == true)

{

SqlCommand commandu = new SqlCommand($"UPDATE Balance set money = (select top 1 money from balance where card = '1') - @aC where id = (select top 1 id from balance where card = '1') AND card = '1'", database.getConnection()); //добавление общей суммы баланса

commandu.Parameters.Add("@aC", SqlDbType.VarChar).Value = AddCashTextBox.Text;

database.OpenConnection();

commandu.ExecuteNonQuery();

f.Update();

}

else

{

SqlCommand commandu = new SqlCommand($"UPDATE Balance set money = (select top 1 money from balance where card = '0') - @aC where id = (select top 1 id from balance where card = '0') AND card = '0'", database.getConnection());

commandu.Parameters.Add("@aC", SqlDbType.VarChar).Value = AddCashTextBox.Text;

database.OpenConnection();

commandu.ExecuteNonQuery();

f.Update();

}

SqlCommand command = new SqlCommand($"INSERT INTO Expenses (Сумма, Категория, Дата) VALUES (@aC, '{cat}',GetDate())", database.getConnection());

command.Parameters.Add("@aC", SqlDbType.VarChar).Value = AddCashTextBox.Text;

database.OpenConnection();

if (command.ExecuteNonQuery() == 1)

MessageBox.Show("Создано!");

else

MessageBox.Show("Не удачно! :(");

database.CloseConnection();

f.Update();

}

private void Window\_Closed(object sender, EventArgs e)

{

f.Update();

}

private void CheckBox\_Checked(object sender, RoutedEventArgs e)

{

}

}

}

//DataBase.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpfProject

{

class Database

{

SqlConnection Connection = new SqlConnection("Data Source=.\\SQLEXPRESS;Initial Catalog=MoneyDesktop;Integrated Security=True"); //подключение к sql

public void OpenConnection()

{

if (Connection.State == System.Data.ConnectionState.Closed)

Connection.Open();

}

public void CloseConnection()

{

if (Connection.State == System.Data.ConnectionState.Open)

Connection.Close();

}

public SqlConnection getConnection()

{

return Connection;

}

}

}

//MainWindow.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfProject

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public class Expenses

{

public int Count { get; set; }

public int Category { get; set; }

public DateTime Date { get; set; }

public Expenses(int count, int category, DateTime date)

{

Count = count;

Category = category;

Date = date;

}

}

public MainWindow()

{

InitializeComponent();

Update();

}

public void Update()

{

Database database = new Database();

database.OpenConnection();

SqlCommand command3 = new SqlCommand("UPDATE Balance set MoneySum = (select sum(money) from balance);", database.getConnection());

using (SqlDataReader reader = command3.ExecuteReader())

{

if (reader.Read())

{

CardSum.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command = new SqlCommand("SELECT sum(Money) from Balance where card = '1'", database.getConnection());

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

CardSum.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command1 = new SqlCommand("SELECT sum(Money) from Balance where card = '0'", database.getConnection());

using (SqlDataReader reader = command1.ExecuteReader())

{

if (reader.Read())

{

CashSum.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command2 = new SqlCommand("SELECT sum(Money) from Balance", database.getConnection());

using (SqlDataReader reader = command2.ExecuteReader())

{

if (reader.Read())

{

SumMoney.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command4 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Продукты'", database.getConnection());

using (SqlDataReader reader = command4.ExecuteReader())

{

if (reader.Read())

{

Label1.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command5 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Мобильная связь'", database.getConnection());

using (SqlDataReader reader = command5.ExecuteReader())

{

if (reader.Read())

{

Label2.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command6 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Досуг'", database.getConnection());

using (SqlDataReader reader = command6.ExecuteReader())

{

if (reader.Read())

{

Label3.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command7 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Здоровье'", database.getConnection());

using (SqlDataReader reader = command7.ExecuteReader())

{

if (reader.Read())

{

Label4.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command8 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Кафе'", database.getConnection());

using (SqlDataReader reader = command8.ExecuteReader())

{

if (reader.Read())

{

Label5.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command9 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Бар'", database.getConnection());

using (SqlDataReader reader = command9.ExecuteReader())

{

if (reader.Read())

{

Label6.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command10 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Одежда'", database.getConnection());

using (SqlDataReader reader = command10.ExecuteReader())

{

if (reader.Read())

{

Label7.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command11 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Автомобиль'", database.getConnection());

using (SqlDataReader reader = command11.ExecuteReader())

{

if (reader.Read())

{

Label8.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command12 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Подарки'", database.getConnection());

using (SqlDataReader reader = command12.ExecuteReader())

{

if (reader.Read())

{

Label9.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlCommand command13 = new SqlCommand("select sum(Сумма) from Expenses where Категория='Магазины'", database.getConnection());

using (SqlDataReader reader = command13.ExecuteReader())

{

if (reader.Read())

{

Label10.Content = reader.GetValue(0);

}

else

{

Debug.WriteLine("Не удачно!");

}

}

SqlDataAdapter flex = new SqlDataAdapter("select Сумма, Категория, Дата from Expenses", database.getConnection());

DataTable dt = new DataTable();

flex.Fill(dt);

DataGrid.ItemsSource = dt.DefaultView;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

AddCardMoney addCardMoney = new AddCardMoney(this);

addCardMoney.Show();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

AddCash addCash = new AddCash(this);

addCash.Show();

}

private void CardSum\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

}

private void Image\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Продукты", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_1(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Мобильная связь", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_2(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Досуг", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_3(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Здоровье", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_4(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Кафе", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_5(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Бар", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_6(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Одежда", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_7(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Автомобиль", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_8(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Подарки", this);

category.Show();

}

private void Image\_MouseLeftButtonDown\_9(object sender, MouseButtonEventArgs e)

{

AddCategory category = new AddCategory("Магазины", this);

category.Show();

}

}

}

//HelpSystem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WpfProject

{

/// <summary>

/// Логика взаимодействия для HelpSystem.xaml

/// </summary>

public partial class HelpSystem : Window

{

public HelpSystem()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}

Приложение Б

(справочная)

Формы выходных документов

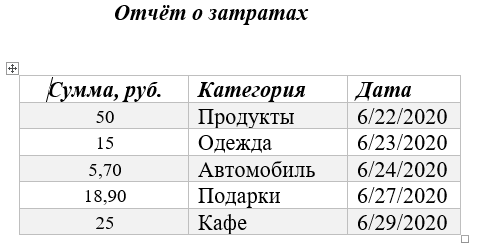


Рисунок Б.1