СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Денежный долг - это достаточно интересная сфера отношений. Мы сами часто даем в долг, у нас просят дать в долг. Все, кто хоть раз участвовал в этом финансовой операции, могут отметить одно правило – взять в долг гораздо легче, чем отдать его.

Ситуаций может быть много, это и друг забывший дома кошелек, и ситуация, когда один человек покупает подарок, а остальные позже с ним рассчитываются, это очень актуальна в такие праздники как: 8 марта и 23 февраля. Или организация мероприятие, где друзья платят друг за друга. Это может быть поход в кино или кофе группы людей. Бывает, коллеги на работе скидываются на пиццу. Совместный отдых друзей. Но что делать если иногда небольшие суммы забываются? Конечно, можно вести записи на бумажке, но а можно использовать мобильное приложение, которое будет вести учет финансовых взаимоотношений пользователей. Будет напоминать о том, что нужно вернуть долг. Не нужно все это вести в голове, а просто посмотреть в приложение.

Объектом разрабатываемое системы являются финансовое взаимоотношение группы людей.

Целью автоматизации является сокращение затрат и времени на учет и обработку финансовых операций группы пользователей.

Система должна обеспечивать:

- ведение базы данных (данные о пользователях, работах, о друзьях пользователях, данные о транзакциях пользователя, долги пользователя, займы пользователя)
- комплекс задач, обеспечивающих управление доступом, сохранность, восстанавливаемость информации, авторизацию пользователей, регистрация пользователей, управление профилями пользователей.
- функционал для поиска пользователей.
- комплекс задач, обеспечивающих управлением транзакциями.
- просмотр займов
- функционал для добавления в друзья, удаления из друзей.
- просмотр личной информации друга, где можно будет увидеть номер карточки, куда перечислить деньги.
- просмотр долгов и займов определённого друга.

Для разработки системы необходимо:

- обследовать предметную область;
- выбрать концепцию построения системы и средств разработки;
- разработать структуру системы, интерфейс, отчетные формы, модули ПО, спроектировать БД;

- провести тестирование, испытание системы, документирование результатов разработки.

1 СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ

1.1 Описание объекта автоматизации

Объектом автоматизации данного дипломного проекта является «финансовое взаимоотношение группы людей.».

Цель автоматизации – сокращение затрат и времени на учет и обработку финансовых операций группы пользователей.

Деньги – это универсальный эквивалент, который определяет стоимость товаров и услуг. Во все времена деньги играли важнейшую роль в повседневной жизни людей. Все мы пользуемся этим ресурсом. Неумелое использования этого ресурса приводит к ситуации, когда нужно одалживать деньги. Также в жизни бывают сложные ситуации, когда нужно что-то купить ценное, к примеру квартиру или машину. Или нужен начальный капитал для того, чтобы открыть свой бизнес. Таких ситуаций в жизни очень много. Да можно эти проблемы решить походом в банк, и взять кредит под большие проценты. И выплачивать этот кредит несколько десятков лет. Но я считаю, что лучше взять в долг у родных или друзей. И потихоньку выплачивать долг без каких-либо процентов.

Рассмотрим и другие жизненные ситуации. Группа людей скидывается на какое-нибудь мероприятие. Это может быть праздник: новый год, 8 марта, 23 февраля и т. д. Или к примеру, друзья спланировали совместный отдых. Поход друзей в кино. Заказ пиццы коллег на работе. Бронирование спортивного зала футбольной команде. Все выше упомянутые ситуации в основном проходят по одному сценарию. Есть человек, который ответственный за мероприятие. Этот ответственный бедолага должен собрать у всех деньги и оплатить за все. Если большая компания людей, то найдутся те кто забудут принести ну или те, у кого нет денег сейчас и они обещают потом отдать. Их нужно всех запомнить и потом еще и напоминать, тратя на это свои нервы.

Сейчас двадцать первый век и деньги можно скинуть через интернет на банковскую карточку. Но для этого нужно знать номер карты одолжившего.

Как видим очень много жизненных ситуаций связанных с долгами. Проблемы, причисленные мной до сих пор актуальны.

И перед собой я поставил цель создать автоматизированную систему. Которая решала бы эти проблемы.

Существует не так уж много систем для учета и управлениям долгами. В, изученных мною, аналогах были выявлены существенные недостатки. И эти недостатки необходимо избежать в своей системе.

В ходе разработки данного проекта были выявлены следующие особенности системы, которые необходимо учесть в данном продукте:

- Хранение бд на веб –сервере а не локально, это позволит восстановит данные при загрузки приложения на новое устройство. При переустановки приложения также все можно будет восстановить.
- Создать приложение, в котором можно реально работать с реальными пользователями, а не приложения формата записная книжка.
- Поиск существующих пользователей, добавления в друзья, удаления из друзей.
- Создание транзакций, и управление ими.
- Просмотр должников.
- Просмотр займов.
- Просмотр личной информации друга, где можно будет увидеть номер карточки, куда перечислить деньги.
- Просмотр долгов и займов определённого друга.

1.2 Обзор аналогичных приложений

На рынке существуют подобные решение, так как данные проблемы актуальные, приведем основные и самые популярные на рынке, на данный момент. Но все они в основном имеет один недостаток, так как они работают как записная книжка и при удалении приложения пропадут и все долги. Конечно, если у тебя долгов больше, чем займов - это даже хорошо. Ну если серьезно - это большой недостаток. А также они работают с записями, а не реальными пользователями. В таких приложения ты не можешь воздействовать на пользователя, напомнить ему, или потвердеть что деньги вы получили. Это тоже большой недостаток.

Книга Долгов – это небольшое приложение, помогающее вести учет долгов и должников, с простым, не загруженным лишними возможностями интерфейсом.

Просто нажмите кнопку "+", когда платите за кого-то, даёте другу деньги в долг или сами берёте в долг. Введите имя человека, выберите "дать" или "взять" в долг, введите сумму - готово! Меняйте валюту при необходимости, выбирая из списка "любимых" или всех валют мира. Измените дату или введите комментарий при желании, для удобства дальнейшего учёта. Введите имена нескольких человек, чтобы разделить долг между ними; добавьте "я" или "себе", чтобы учесть себя в разделении затрат.

Чтобы погасить долг, проведите пальцем по строке в списке долгов - или используйте действия, такие как "Погасить этот долг" или "Погасить полностью", чтобы сохранить всю историю обмена. Ее можно будет отправить другу или скопировать, используя действие "Отправить".

Смотри рисунок 1.1.

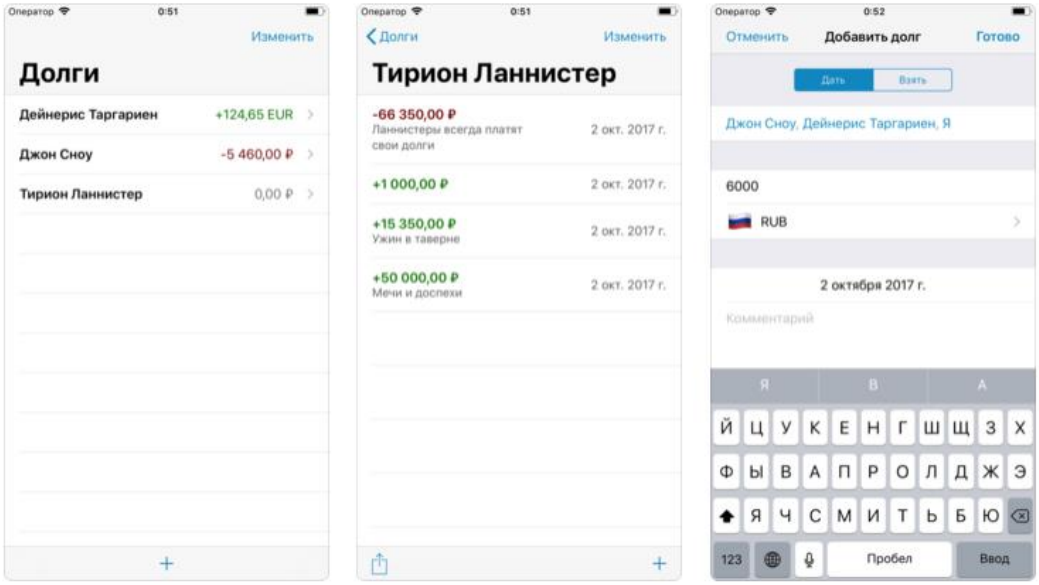


Рисунок 1.1 – Приложение - книга долгов

IOU - приложение для учета долгов, как денежных, так и предметов. С такими функциями как оповещения, регулярные долги и платежи вы сможете вести учет займов, периодических платежей и личных долгов - все в одном приложении! Синхронизация с ioutool.net позволяет хранить и редактировать данные на разных мобильных устройствах и в браузере. Вы даже можете предоставить совместный доступ к долгам вашим контактам. Скрин приложения изображён на рисунке 1.2.

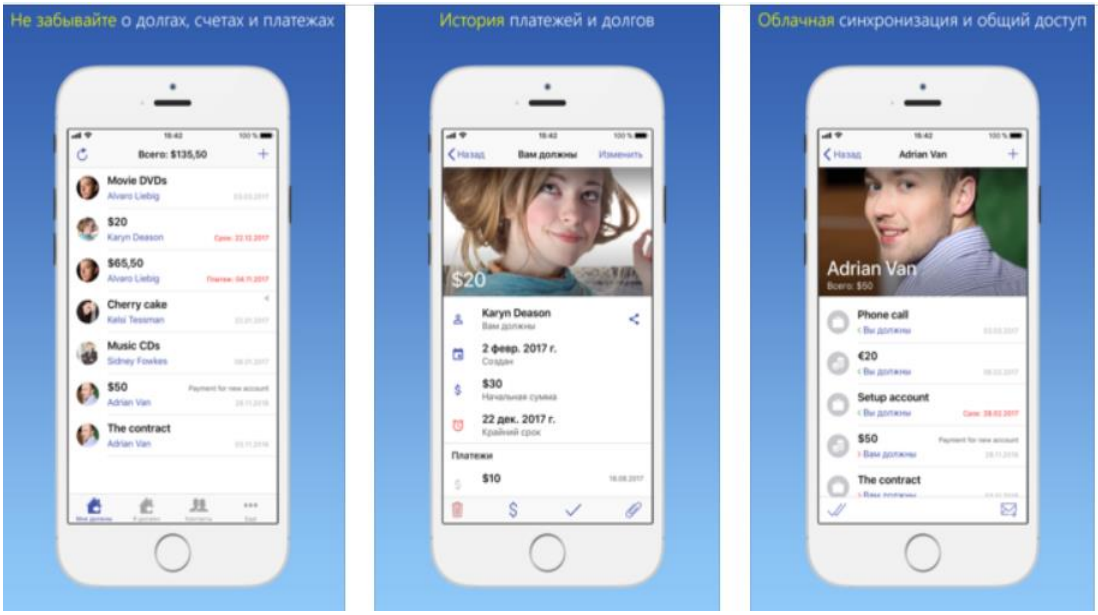


Рисунок 1.2 – Приложение - IOU

PayMeBack — очень простое приложение, несущее всего лишь одну функцию. Как вы уже могли догадаться, мы будем пользоваться им для того, чтобы записать,

когда вы берете в долг или одалживаете кому-либо. Скрин приложения изображён на рисунке 1.3.

Рисунок 1.3 – Приложение – PayMeBack

Менеджер долгов разработанное, как для отдельных, так и для корпоративных пользователей приложение "Менеджер долгов" - отличный инструмент для отслеживания занятых и одолженных сумм (дебиторы/кредиторы).

Многофункциональное и универсальное приложение "Менеджер долгов" может использоваться для разнообразных видов финансовой деятельности, включая коммерческую деятельность (например, продажу товаров и услуг), крупные займы, равноправное кредитование, долговые обязательства, микрофинансирование (микрокредитование) и любые другие случаи, когда необходимо отслеживание и управление денежными ресурсами.

Скрин приложения изображён на рисунке 1.4.

Стандартные возможности:

• Отслеживание средств, причитающихся вам/с вас

• Группировка по должникам/кредиторам/физическим лицам

• Поддержка различных валют

• Календарные записи и напоминания
• История транзакций
• Сведения о долгах по валюте

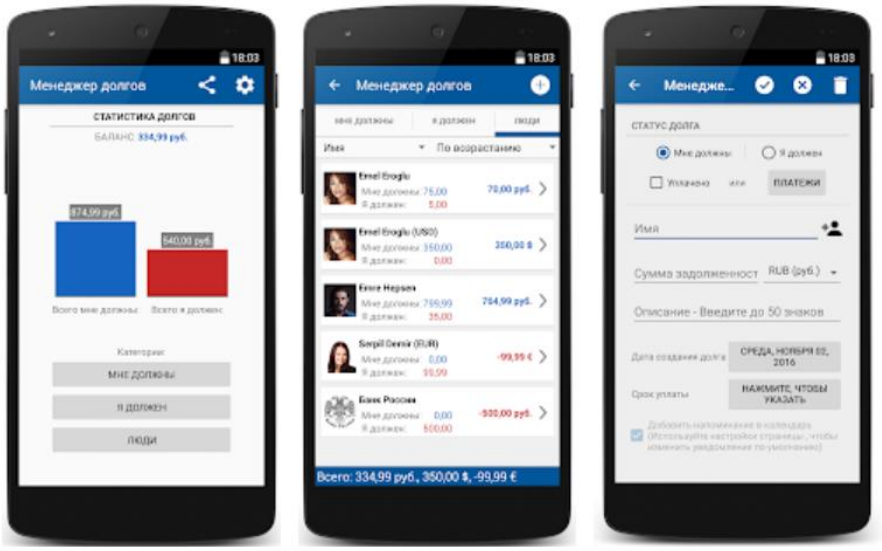


Рисунок 1.4 – Приложение – Менеджер долгов

Учет долгов - это приложение очень простое, есть всего два раздела: Мне должен и Я должен.

Программа также считает баланс долгов – сколько вы должны и сколько вам должны по каждой валюте. Скрин приложения изображён на рисунке 1.5.

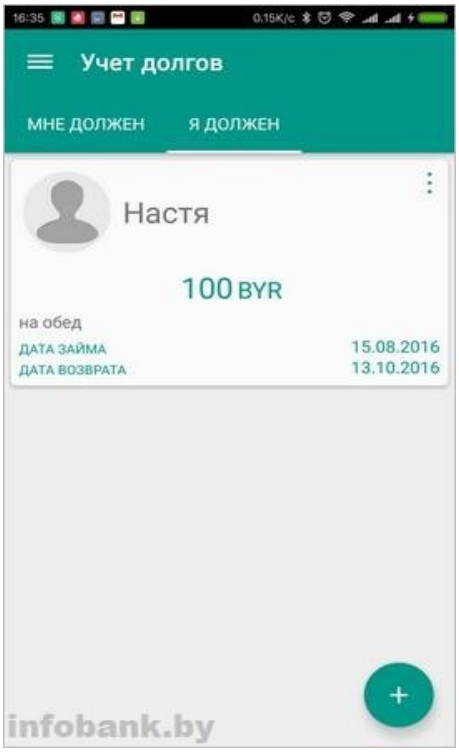


Рисунок 1.5 – Приложение – Учет долгов

1.3 Обоснование необходимости разработки системы

Все выше перечисленные приложения разработаны для IOS и android. Я же поставил перед собой цель разработать uwp-приложение (universal windows platform. Данное приложение поддерживается широким кругом устройств на которых установлена Windows 10. Это десктопы, планшеты, смартфоны, большие планшеты Surface Hub. В итоге мы получим приложение, которое может работать на всех вышеперечисленных устройствах.

Некоторые аналоги никак не взаимодействуют с веб-сервером и это - очень большой недостаток. При удалении приложения все долги теряются и их невозможно восстановить. Все они работают локально и подобны записной книги, которая всегда под рукой. Но сегодня нет проблемы с доступом к интернету. И я считаю, что такого рода приложения должны обеспечивать возможность – восстановить данные.

В рассмотренных аналогах работают с записями, а не реальными пользователями. В таких приложения ты не можешь воздействовать на пользователя, напомнить ему, или потвердеть что деньги вы получили. Это тоже большой недостаток. Реальный пользователь имеют личную информацию: номер банковской карты, номер телефона, адрес электронной почты. И посмотрев в приложении номер карточки, у тебя появляется возможность скинуть через интернет-банкинг, не выходя из дома, что очень комфортно.

В большинстве приложениях не предусмотрено добавление транзакциях для группы пользователей. А это функционал, по моему мнению, является очень полезным. В пункте 1.1 дипломного проекта были рассмотрено множество проблем, которые решаются этим функционалом.

Как видим аналоги имеют множество недостатков. Этим и обусловлена необходимость создание данной системы.

1.4 Постановка задачи на создание системы

Объектом разрабатываемое системы являются финансовое взаимоотношение группы людей.

Целью автоматизации является сокращение затрат и времени на учет и обработку финансовых операций группы пользователей.

Система должна обеспечивать:

- ведение базы данных (данные о пользователях, работах, о друзьях пользователях, данные о транзакциях пользователя, долги пользователя, займы пользователя)

- комплекс задач, обеспечивающих управление доступом, сохранность, восстанавливаемость информации, авторизацию пользователей, регистрация пользователей, управление профилями пользователей.

- функционал для поиска пользователей.

- комплекс задач, обеспечивающих управлением транзакциями.

- просмотр займов

- функционал для добавления в друзья, удаления из друзей.

- просмотр личной информации друга, где можно будет увидеть номер карточки, куда перечислить деньги.

- просмотр долгов и займов определённого друга.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Проектирование структуры системы

Общая структура системы изображена на рисунке 2.1

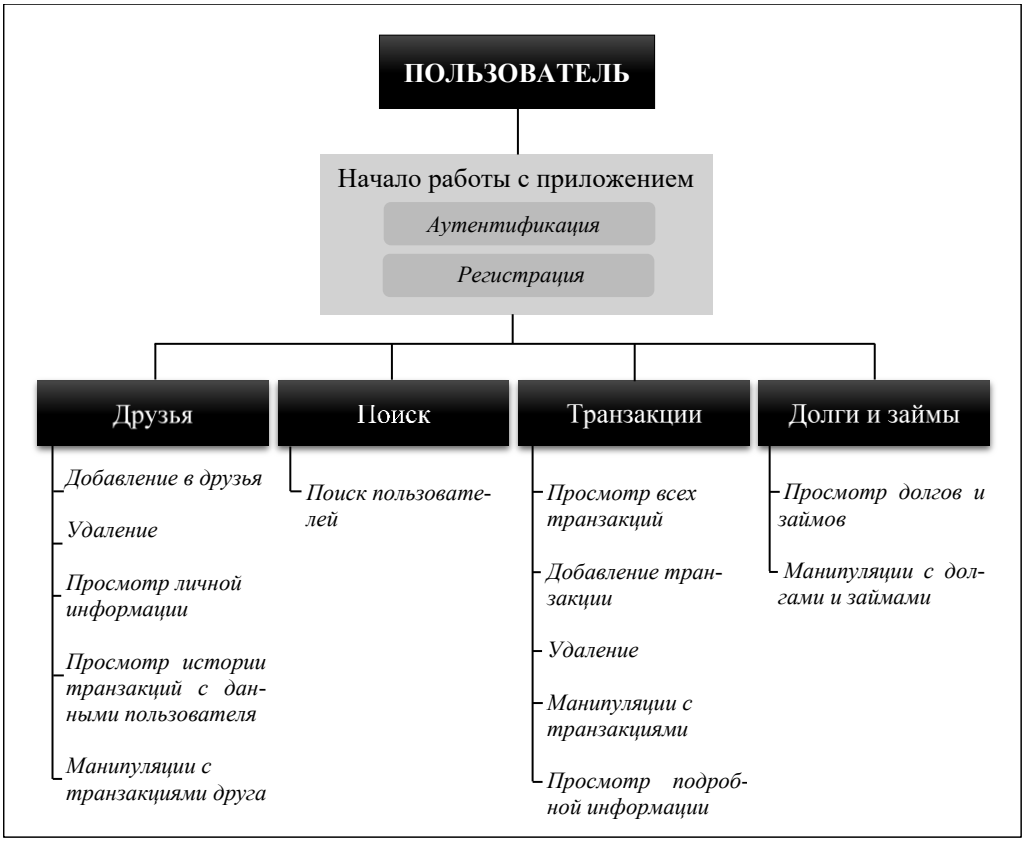


Рисунок 2.1 – Схема системы

Я выбрал архитектуру MVVM для разработки программы. Так как MVVM используется для разделения модели и её представления, что необходимо для изменения их отдельно друг от друга. Например, разработчик задает логику работы с данными, а дизайнер соответственно работает с пользовательским интерфейсом.

Теоретические сведение об архитектуре:

Шаблон Model-View-ViewModel (MVVM) — применяется при проектировании архитектуры приложения. Первоначально был представлен сообществу Джоном Госсманом (John Gossman) в 2005 году как модификация шаблона Presentation Model. MVVM ориентирован на современные платформы разработки, такие как Windows Presentation Foundation, Silverlight от компании Microsoft, ZK framework.

Шаблон MVVM делится на три части:

1) Модель (англ. Model), так же, как в классической MVC, Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

2) Представление (англ. View) — это графический интерфейс, то есть окно, кнопки и т. п. Представление является подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновленное значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.

3) Модель Представления (англ. ViewModel) является, с одной стороны, абстракцией Представления, а с другой, предоставляет обёртку данных из Модели, которые подлежат связыванию. То есть, она содержит Модель, которая преобразована к Представлению, а также содержит в себе команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Общая схема архитектуры MVVM изображена на рисунке 2.2

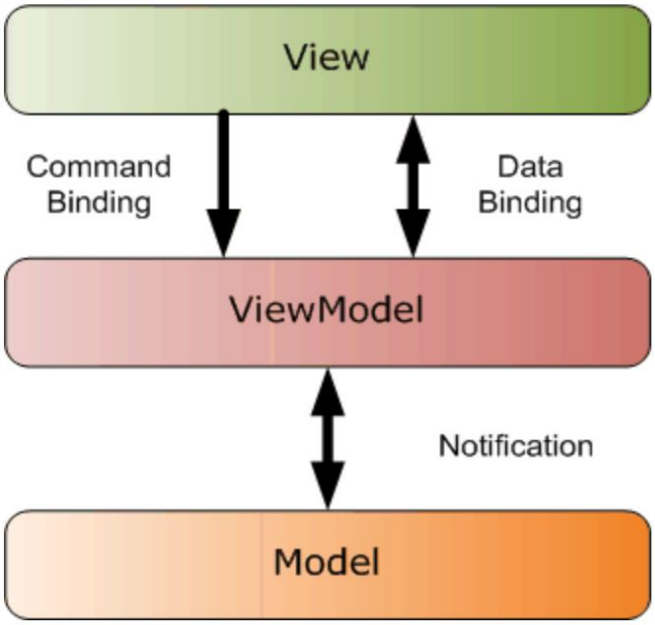


Рисунок 2.2 – Схема архитектуры MVVM.

2.2 Проектирование программного обеспечения системы

Организация корневого каталога проекта. Корневой каталог исходного кода системы представлен на рисунке 2.3.

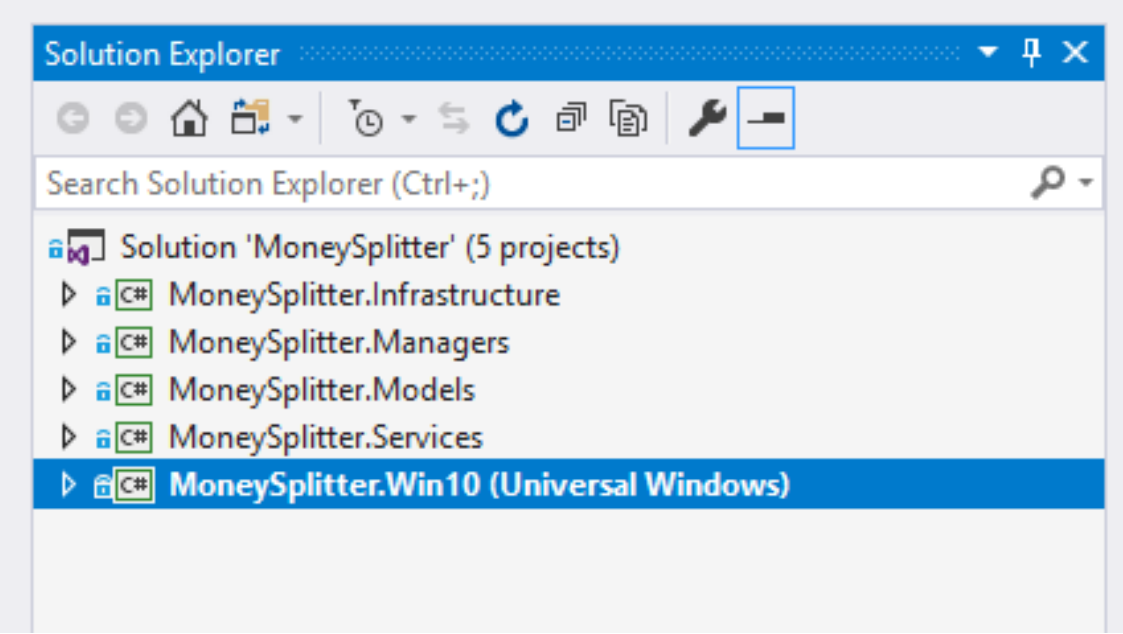


Рисунок 2.3 - Корневой каталог проекта

Проект будет состоять из 5 модулей. Каждый модель он не зависим и в последующим его можно заменить другим.

Рассмотрим назначение модулей:

Infrastrucure - модуль в котором хранятся все интерфейсы программы. Интерфейсы позволяют определить некоторый функционал для класса, не имеющий конкретной реализации. Класс, который подписан на конкретный интерфейс, обязан реализовать весь функционал, прописанный в интерфейсе. На один интерфейс могут порисоваться множество классов. И интерфейс в программе может служить, как обобщённый тип. Но работать мы будем с классами реализующие этот интерфейс.

Managers модуль, где реализуются менеджеры. Менеджеры — это такие классы способны управлять и манипулировать данными. Также они умеют обрабатывать входные данные.

Models каталог, содержащий пользовательские классы, используемые в приложении, все они описывают сущности.

Services модуль, где реализуются сервисы. Сервисы это классы, которые предоставляют данные. Они никак не манипулируют с данными, а про сто предоставляют ее. Есть сервисы, которые запрашивают данные у веб-сервера, и предоставляют её в удобном виде. Такие сервисы, называются api-сервисы.

Win10 это модуль отвечающий за клиентскую часть приложения. Все остальные модули вспомогательные. Они кроссплатформенные, ни как как не зависят от клиентской части. В теории, мы можем по-разному реализовать клиент. И это не скажется на всех остальных модулях.

На рисунке 2.4 предоставлен каталог Win10.

В данном части приложение осуществляется работа с пользовательским интерфейсом. Рисуются представления, разрабатываются стили. И создаются функционал для вывода необходимой информации на экран.

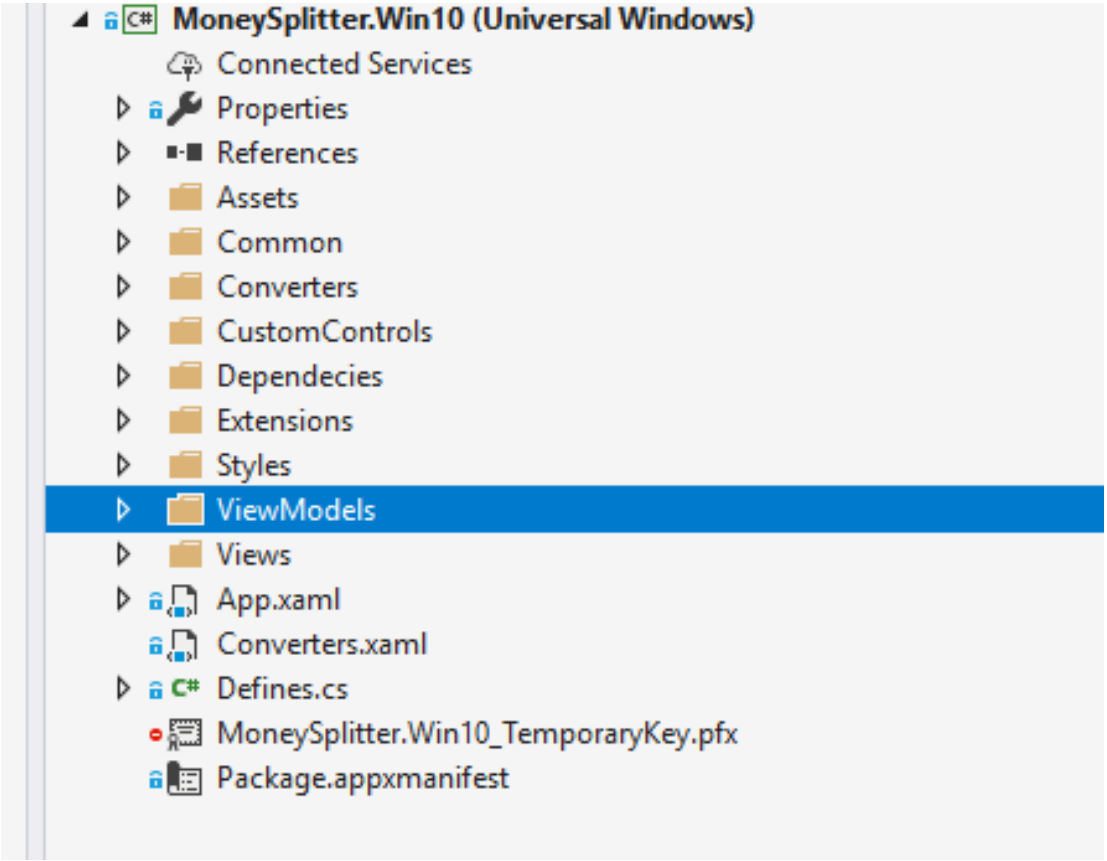


Рисунок 2.4 – Содержание каталога Win10

2.3 Структура информационного обеспечения

Информационное обеспечение приложения будет представлять собой базу данных. База данных будет содержать семь сущностей (таблиц). В логическую модель базы данных включены такие таблицы, как:

- пользователь (User);
- транзакция (Transaction);
- друга транзакция (FriendTransaction);
- событие транзакции (TransactionEvent);
- участник транзакции (CollaboratorTransaction);
- участник, который отдал деньги, (CollaboratorInFinished);
- участник, который собирается отдать деньги (CollaboratorInProgress).

Логическая модель системы изображена на рисунке 2.5.

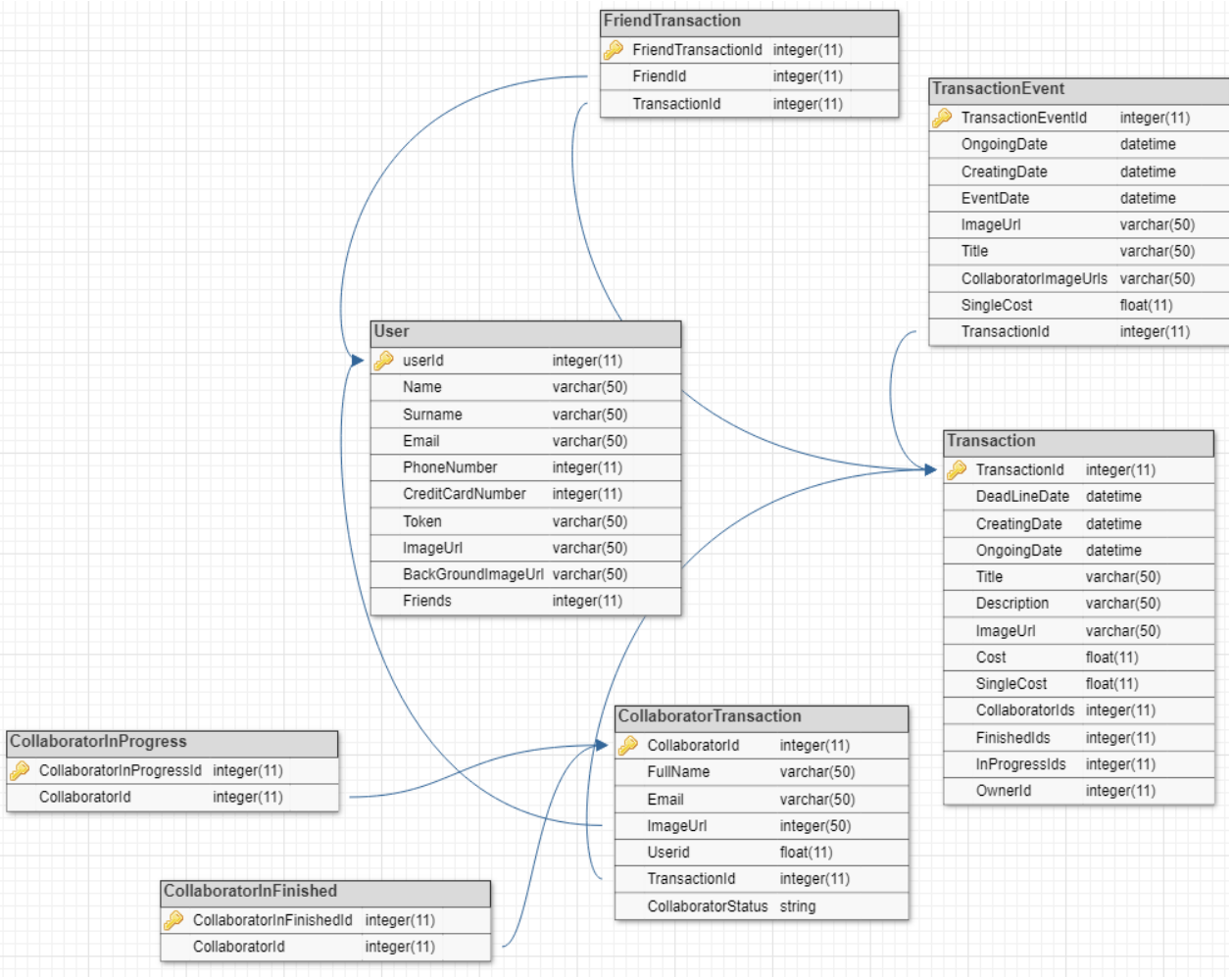


Рисунок 2.5 – Логическая модель системы

Свойства сущности выступают в роли атрибутов в базе данных, они представлены в таблице описания атрибутов по сущности administrators (таблица 2.2).

Таблица 2.2 – Описание атрибутов по сущности administrators

Имя сущности	Имя атрибута	Назначение атрибута	Тип данных
user	UserId	Уникальный идентификатор	int
	Name	Имя	varchar(50)
	Surname	Фамилия	varchar(50)
	Email	Электронная почта	varchar(50)
	PhoneNumber	Номер телефона	int
	CreditCardNumber	Номер банковской карты	int
	Token	Идентификатор доступа	varchar(50)
	ImageUrl	Ссылка на автатарку	varchar(50)
	BackGroundUrl	Ссылка на картинку фона	varchar(50)
	Friends	Друзья	varchar(50)

Описание сущности transaction представлено в таблице 2.3.

Таблица 2.3 – Описание сущности transaction

Имя сущности	Описание сущности	Псевдонимы	Особенности использования
transaction	Объект, содержащий информацию о транзакциях: id, название, описание, стоимость участники и т.д.	Транзакция	Используется для хранения и предоставления данных о транзакции.

Свойства сущности выступают в роли атрибутов в базе данных, они представлены в таблице описания атрибутов по сущности transaction (таблица 2.4).

Таблица 2.4 – Описание атрибутов по сущности exams

Имя сущности	Имя атрибута	Назначение атрибута	Тип данных
transaction	TransactionId	Уникальный идентификатор	int
	Title	Название	varchar(50)
	Description	Описание	varchar(50)
	OwnerId	Владелец транзакции	int
	ImageUrl	Изображение транзакции	varchar(50)
	Cost	Стоимость	float
	DeadLineDate	-	date
	CreatingDate	Дата создания	date
	OngoingDate	Дата события	date
	CollaboratorIds	Участники	int
	InProgressIds	-	int
	FinishedIds	-	int
	SingleCost	-	float

Описание сущности collaboratorTransaction представлено в таблице 2.5.

Свойства сущности выступают в роли атрибутов в базе данных, они представлены в таблице описания атрибутов по сущности collaboratorTransaction (таблица 2.6).

Описание сущности TransactionEvent представлено в таблице 2.7.

Свойства сущности выступают в роли атрибутов в базе данных, они представлены в таблице описания атрибутов по сущности TransactionEvent представлено в таблице 2.8

Описание сущности FriendTransaction представлено в таблице 2.9.

Свойства сущности выступают в роли атрибутов в базе данных, они представлены в таблице описания атрибутов по сущности FriendTransaction (таблица 2.10).

Таблица 2.5 – Описание сущности collaboratorTransaction

Имя сущности	Описание сущности	Псевдонимы	Особенности использования
CollaboratorTransaction	Объект, содержащий информацию о участнике: id, полное имя, транзакция, статус участника, электронная почта и.т.д.	Участник транзакции	Используется для хранения и предоставления данных о участниках транзакции

Таблица 2.6 – Описание атрибутов по сущности exam_category

Имя сущности	Имя атрибута	Назначение атрибута	Тип данных
CollaboratorTra-nsaction	collaboratorId	Уникальный идентификатор	int(11)
	FullName	Полное имя	varchar(50)
	Email	Электронная почта	varchar(50)
	Cost	Стоимость	varchar(50)
	ImageUrl	Изображение участника	varchar(50)
	TransactionId	Транзакция	int
	CollaboratorStatus	Участника статус	varchar(50)
	UserId	Пользователь	int

Таблица 2.7 – Описание сущности TransactionEvent

Имя сущности	Описание сущности	Псевдонимы	Особенности использования
TransactionEvent	Объект, содержащий информацию о событии транзакции: id, название, стоимость дата события, изображение и.т.д.	Событие транзакции	Используется для хранения и предоставления данных о событии транзакции

Таблица 2.8 – Описание атрибутов по сущности questions

Имя сущности	Имя атрибута	Назначение атрибута	Тип данных
TransactionEvent	TransactionEventId	Уникальный идентификатор	int(20)
	Title	Название	varchar(50)
	ImageUrl	Изображение транзакции	varchar(50)

Продолжение таблицы 2.8

TransactionEvent	TransactionEventId	Уникальный идентификатор	int(20)
	EventDate	Дата события	date
	CreatingDate	Дата создания	date
	OngoingDate	Дата события	date
	CollaboratorImage	Изображение участников	blob
	TransactionId	Транзакция	int
	SingleCost	-	float

Таблица 2.9 – Описание сущности FriendTransaction

Имя сущности	Описание сущности	Псевдонимы	Особенности использования
FriendTransaction	Объект, содержащий информацию транзакциях друга:id, друг, транзакция.	Транзакция друга	Используется для хранения и предоставления данных о транзакции друга.

Таблица 2.10 – Описание атрибутов по сущности FriendTransaction

Имя сущности	Имя атрибута	Назначение атрибута	Тип данных
FriendTransaction	FriendTransactionId	Уникальный идентификатор	int(20)
	TransactionId	Транзакция	int
	FriendId	Друг	int

Описание сущности CollaboratorInProgress представлено в таблице 2.11.

Таблица 2.11 – Описание сущности CollaboratorInProgress

Имя сущности	Описание сущности	Псевдонимы	Особенности использования
CollaboratorInProgress	Объект, содержащий информацию о участниках, которые еще не завершили транзакцию, id, участник транзакции.	Участники, которые еще не завершили транзакцию	Используется для хранения и предоставления данных о участниках, которые еще не завершили транзакцию.

Свойства сущности выступают в роли атрибутов в базе данных

Они представлены в таблице описания атрибутов по сущности CollaboratorInProgress (таблица 2.12).

Таблица 2.12 – Описание атрибутов по сущности CollaboratorInProgress

Имя сущности	Имя атрибута	Назначение атрибута	Тип данных
CollaboratorInProgress	CollaboratorInProgressId	Уникальный идентификатор	int
	CollaboratorId	Участник транзакции	int

Описание сущности CollaboratorInFinished в таблице 2.13.

Таблица 2.13 – Описание сущности CollaboratorInFinished

Имя сущности	Описание сущности	Псевдонимы	Особенности использования
CollaboratorInFinished	Объект, содержащий информацию о участниках, которые завершили транзакцию, id, участник транзакции.	Участники, которые завершили транзакцию	Используется для хранения и предоставления данных о участниках, которые завершили транзакцию.

Свойства сущности выступают в роли атрибутов в базе данных, они представлены в таблице описания атрибутов по сущности CollaboratorInFinished (таблица 2.14).

Таблица 2.14 – Описание атрибутов по сущности users

Имя сущности	Имя атрибута	Назначение атрибута	Тип данных
CollaboratorInFinished	CollaboratorInFinishedId	Уникальный идентификатор	int
	CollaboratorId	Участник транзакции	int

Описание ключей представлено в таблице 2.15.

Таблица 2.15 – Описание ключей

Имя сущности	Первичный ключ	Вспомогательный ключ
пользователь	userId	-
транзакция	transactionId	userId
друга транзакция	friendTransactionId	userId, transactionId
событие транзакции	transactionEventId	transactionId
участник транзакции	CollaboratorId	userId, transactionId

Продолжение таблицы 2.15

Имя сущности	Первичный ключ	Вспомогательный ключ
участник, который отдал деньги,	CollaboratorInFinishedId	CollaboratorId
участник, который собирается отдать деньги	CollaboratorInProgressId	CollaboratorId

2.4 Структура пользовательского интерфейса

В пользовательской части приложения предполагается наличие пользовательского интерфейса, посредством которого пользователи будут взаимодействовать с системой.

При открытии приложения будет открываться страница входа/регистрации. На данной странице расположены две кнопки: «Вход» и «Регистрация». По умолчанию будет доступна форма для входа. На которой присутствуют поля: «Email», «Пароль» и кнопка «Войти». Прототип этой страницы изображен на рисунке 2.4.

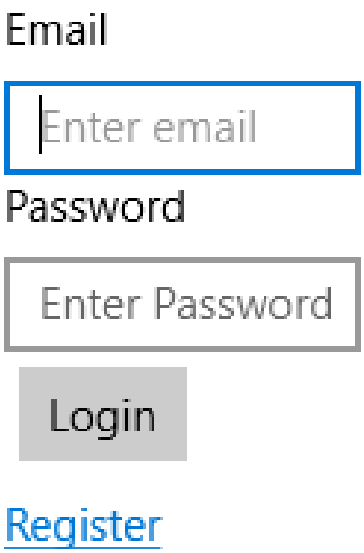


Рисунок 2.6 – Форма входа

При нажатии на кнопку «Регистрация» форма входа будет заменятся на форму регистрации. На которой присутствуют поля: «Email», «Password», «Confirm password», «Name», «Surname», «Phone number», «Browse avatar image» , «Browse background image» и кнопка «Registered». Прототип этой страницы изображен на рисунке 2.7.

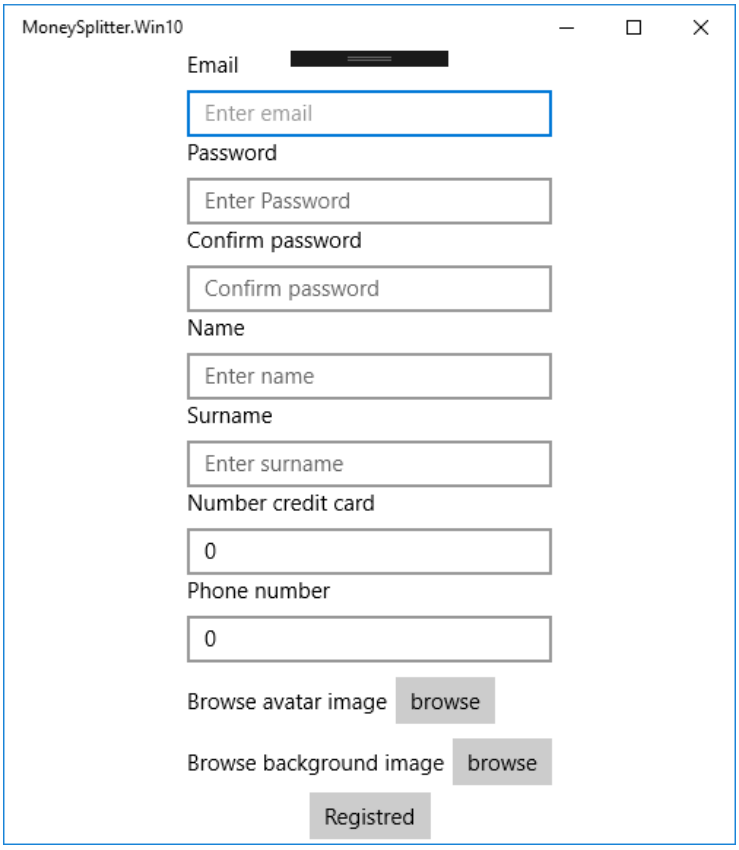


Рисунок 2.7 – Форма регистрации

После входа в систему пользователь попадает домашнею страницу, где есть возможность просмотреть ближайшее событие, а также краткую сводку, сколько вам должны, вы должны. Домашняя страница пользователя изображена рисунке 2.8.

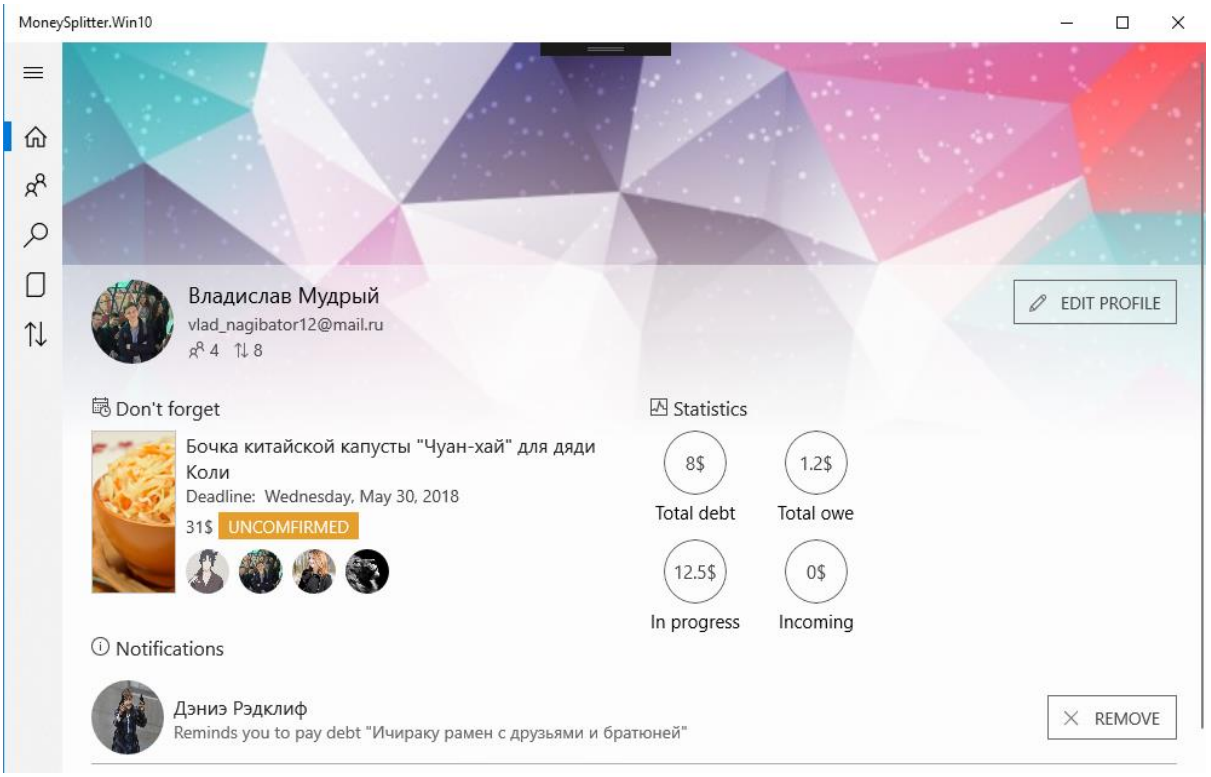


Рисунок 2.8 – Домашняя страница пользователя

Слева будет находится меню, которое имеет три режима, полностью свернутое (для мобильных телефонов), частично свернутое (для планшетов) и полностью развернутое (для компьютеров). При выборе пункта «Friends» пользователь попадает на страницу, где представлены его друзья. Можно узнать подробную информацию друга, щелкнув на иконку друга. Также можно удалить друга, нажав на кнопку «Remove». Прототип этой страницы изображен на рисунке 2.9.

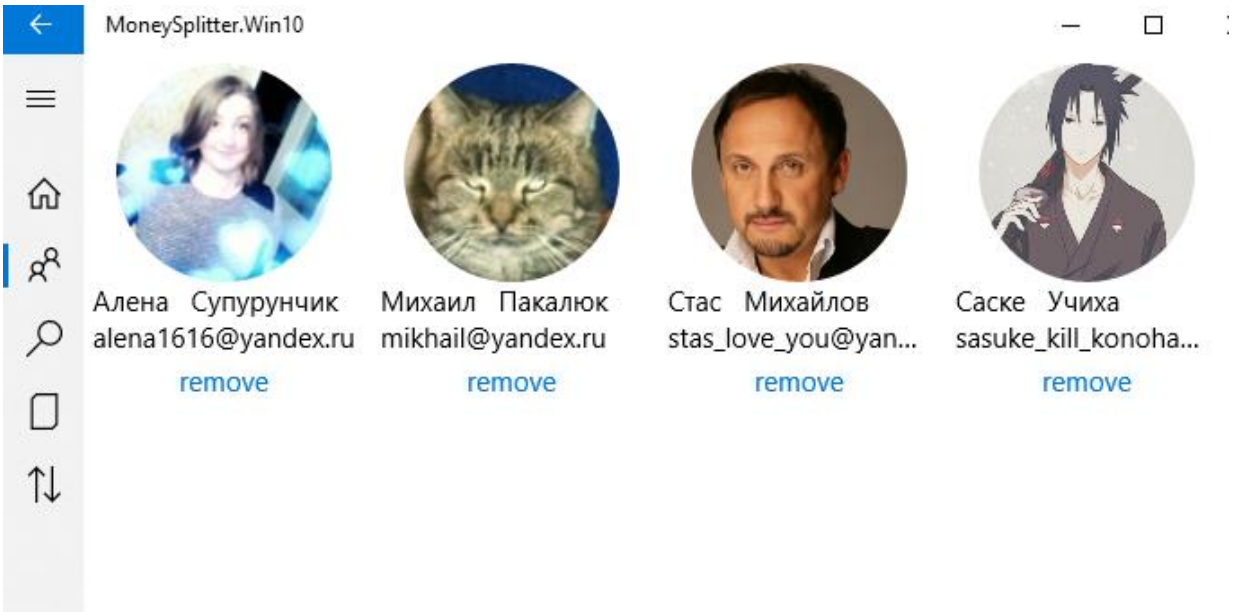


Рисунок 2.9 – Друзья пользователя

Когда пользователь нажимает на и иконку друга он переходит на страницу детали друга. Там он увидит необходимую информацию, чтобы связаться с данным пользователем, это телефон и адрес электронной почты. А также имеется номер банковской карты, куда можно скинуть денежные средства. Справа подсчитана разница сумм всех долгов и займов, и можно увидеть сколько ты должен денег или сколько тебе должен друг. Пользователь имеет право погасить все сделки, если друг в жизни отдал эту сумму. Для этого нужно нажать на кнопку «Approve all».

Внизу страницы показаны следующие транзакции по отношению к пользователю, который зашел на данную страницу: долги друга, займы друга, транзакции в которых вы с другом участвуете и друг другу ничего не должны. Этим с данными транзакциями можно манипулировать использую кнопки справа от транзакций. Прототип этой страницы изображен на рисунке 2.10.

После нажатия на кнопку «Search» в меню, пользователь попадет непосредственно на страницу поиска пользователей. На странице будет располагаться форма для поискового запроса. Вводя строку, мгновенна будут отображаться найденные пользователи. Далее их можно добывать в друзья. Прототип этой страницы изображен на рисунке 2.11.

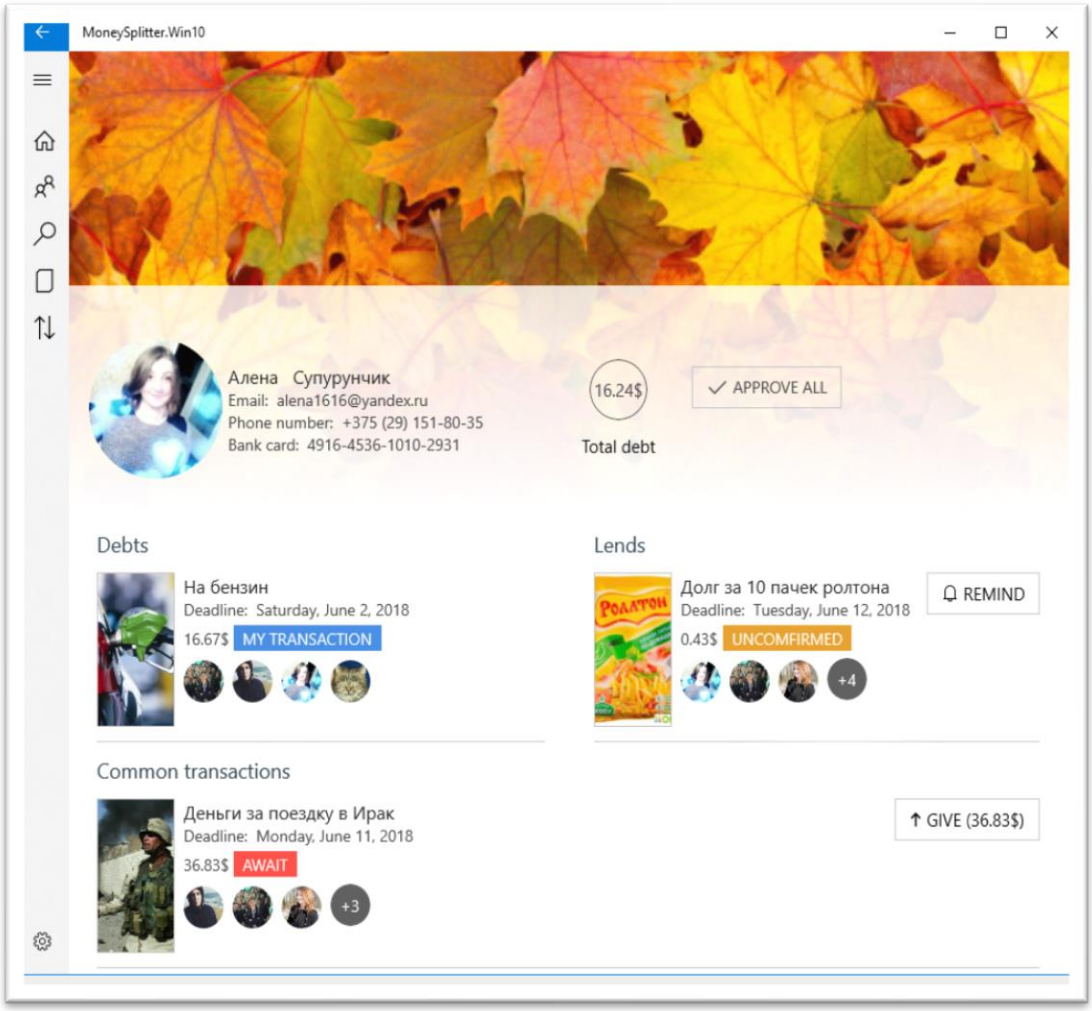


Рисунок 2.10– Детали друга

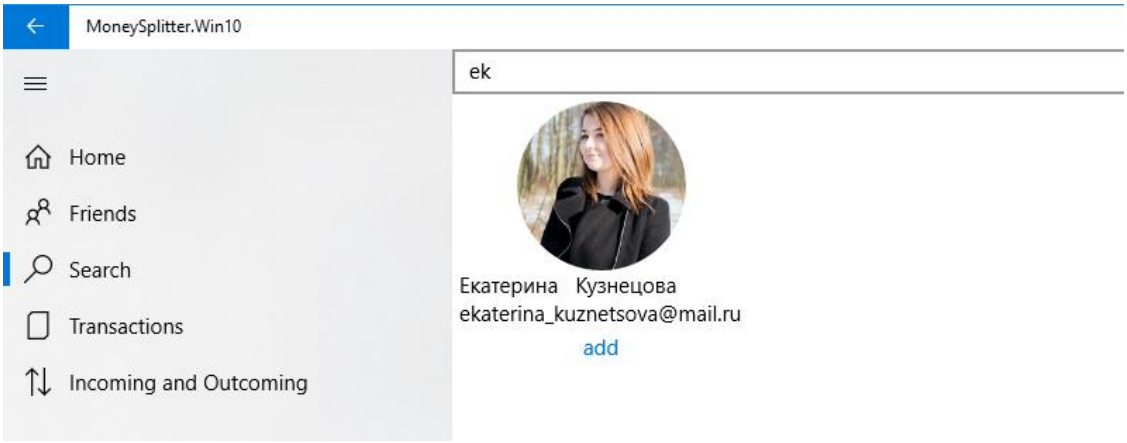


Рисунок 2.11 – Поиск пользователей

После нажатия на кнопку «Transactions» в меню, пользователь попадет на страницу транзакций пользователей. На странице будет располагаться список всех транзакций пользователя. У каждой транзакция хранит следующую информацию. Название, дедлайн или дата события, стоимость, и метка определяющую роль

пользователя в этой транзакции. А также некоторые транзакции имеет действии: отдать, напомнить. Вверху страницы есть кнопка «Add», при нажатии переходим на страницу добавлении транзакции. Также есть кнопка для сортировки по категории. Прототип этой страницы изображен на рисунке 2. 12.

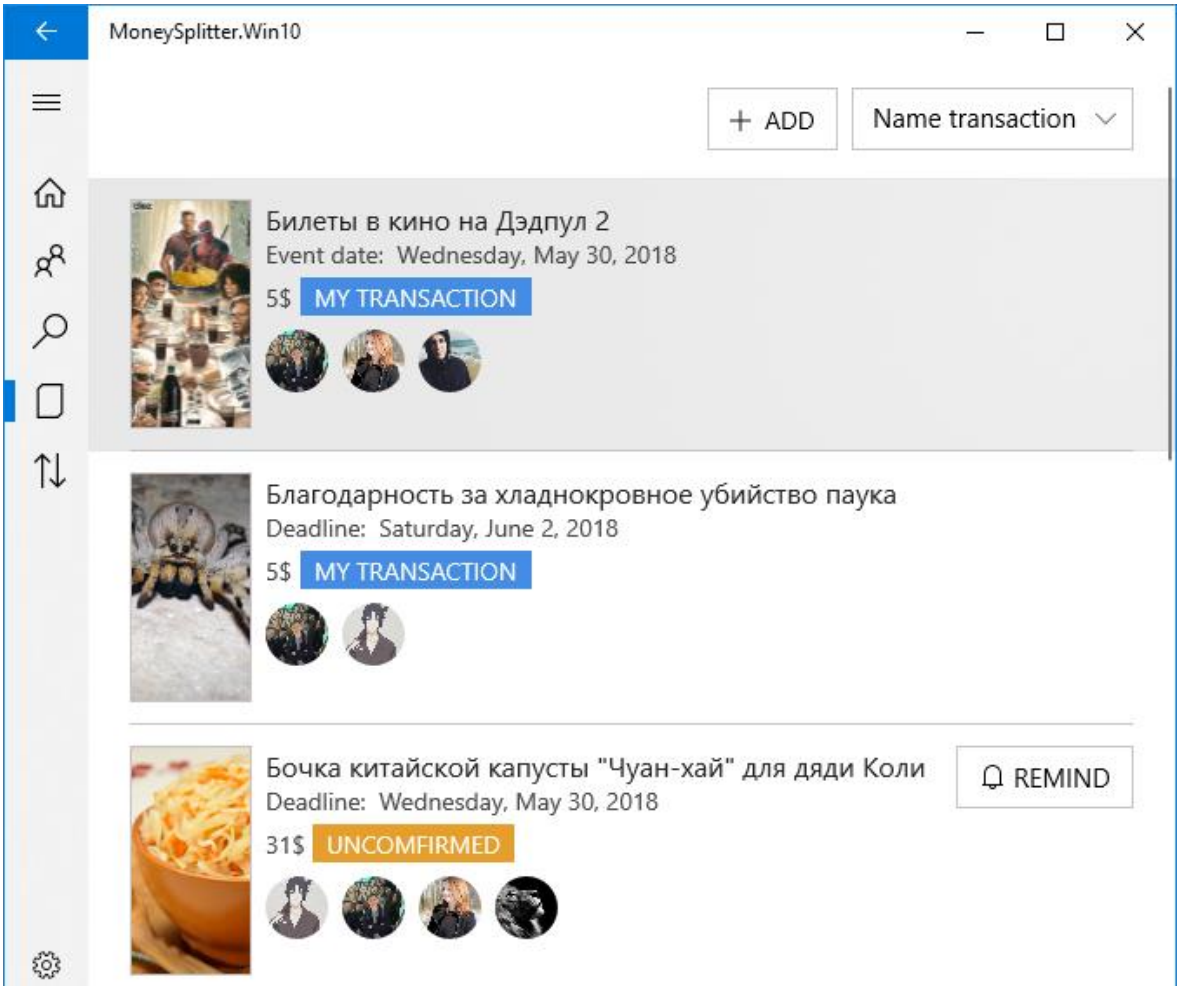


Рисунок 2.12 – Транзакции пользователя

При выборе транзакции открывается детали транзакции. Да данной странице располагается подробная информация о транзакции. Небольшая информация об участниках транзакции, также имеется информация о владельце и прогрессе пользователей в данной сделке. Прототип этой страницы изображен на рисунке 2. 13.

При выборе пункта «Add» пользователь попадает на страницу для создание новой транзакции: имеются поля название, описание цена, дедлайн, изображение, и возможность выбора участников из друзей. Прототип этой страницы изображен на рисунке 2. 14.

После нажатия на кнопку «IncomingAndOutgoing» в меню, пользователь попадет на страницу с должниками и люди, одолжившие пользователю. Показывает кто пользователю должен сколько, нажав на запись, происходит переход на детали пользователя. Прототип этой страницы изображен на рисунке 2. 15

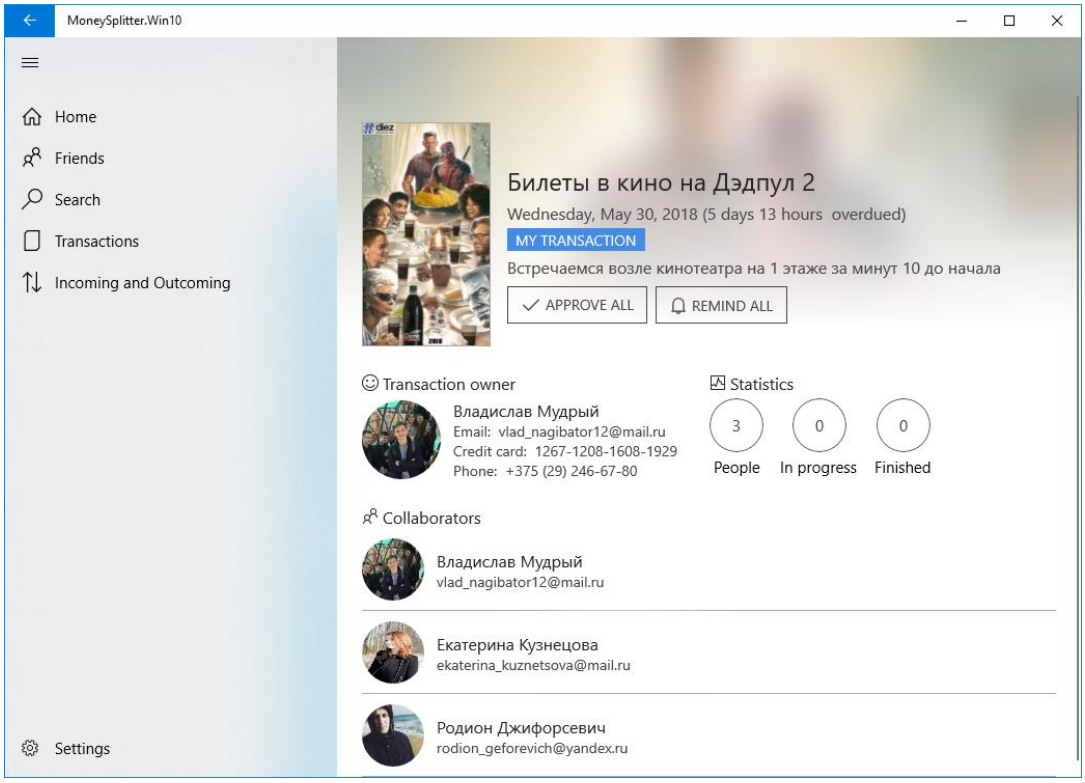


Рисунок 2.13 – Детали транзакции

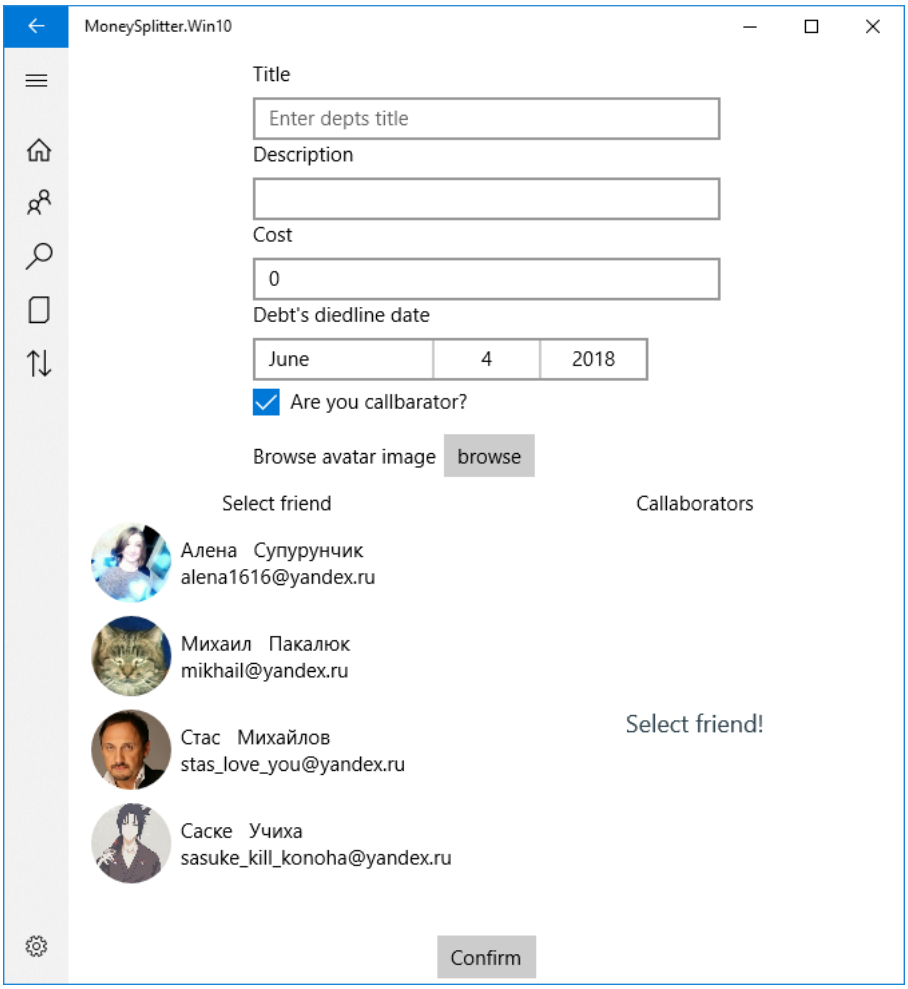


Рисунок 2.14 – Добавление транзакции

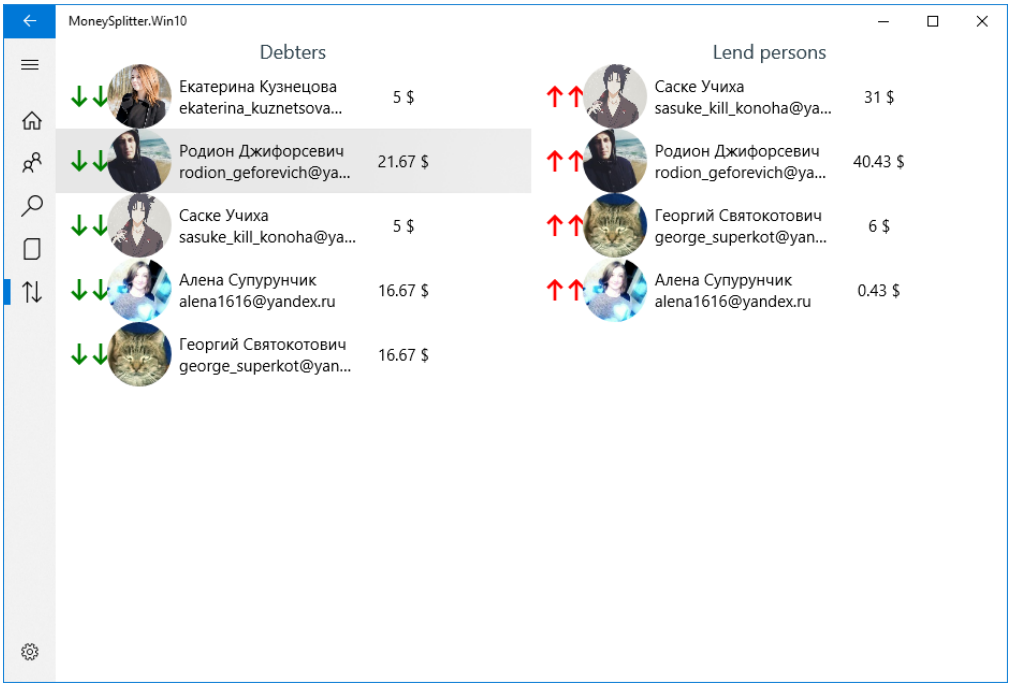


Рисунок 2.15 – Должники и одолжившие

3 РЕАЛИЗАЦИЯ И ИСПЫТАНИЕ

3.1 Выбор средств реализации системы

Для разработки приложения я выбрал язык C# и технологию uwp.

UWP (Universal Windows Platform) представляет собой унифицированную платформу для создания и запуска приложений в Windows 10 и Windows 10 Mobile.

UWP стала результатом фолюции более ранних технологий. Так, с выходом Windows 8 была внедрена новая архитектурная платформа для приложений - Windows Runtime (WinRT), которая позволяла запускать приложения в так называемом режиме Modern (Metro) на десктопах, планшетах. Затем с выходом Windows 8.1 и Windows Phone 8.1 эта технология получила развитие - появились "универсальные приложения", которые можно было запускать сразу Windows 8.1 и WP8.1. И в июле 2015 года официально вышла новая ОС Windows 10. Она использует платформу UWP, которая представляет собой развитие Windows Runtime.

Как подсказывает название платформы, она является универсальной - универсальной для всех устройств экосистемы Windows 10. А это обычные дестопы, планшеты, мобильные устройства, устройства IoT (интернет вещей), Xbox, устройства Surface Hub. И приложение UWP может одинаково работать на всех этих платформах, если на них установлена Windows 10.

Программирование под UWP несет ряд преимуществ:

1)Широта распространения. На текущий момент (апрель 2017) Windows 10 установлена уже более чем на 400 миллионах устройств. На десктопах Windows 10 уже опередила Windows 8/8.1.

2)Поддержка широкого круга устройств. Десктопы, планшеты, смартфоны, большие планшеты Surface Hub, различные IoT-устройства, в перспективе устройства виртуальной реальности HoloLens - круг устрйоств, на которых может работать Windows 10 действительно широк.

3)Поддержка разных языков и технологий программирования. UWP-приложения можно создавать с помощью таких языков, как Visual C++, C#, Visual Basic, JavaScript. В качестве технологии для создания графического интерфейса Visual C++, C# и Visual Basic используют XAML, JavaScript применяет HTML. Кроме того, C++ может вместо XAML использовать DirectX. То есть достаточно распространенные и и знакомые многим технологии.

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык

разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

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.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

XAML (англ. eXtensible Application Markup Language) — расширяемый язык разметки для приложений (произносится [замл] или [зэмл]) — основанный на XML язык разметки для декларативного программирования приложений, разработанный Microsoft.

Модель приложений Vista включает объект Application. Его набор свойств, методов и событий позволяет объединить веб-документы в связанное приложение. Объект Application контролирует выполнение программы и генерирует события для пользовательского кода. Документы приложения пишутся на XAML. Впрочем, с помощью XAML описывается, прежде всего, пользовательский интерфейс. Логика приложения по-прежнему управляется процедурным кодом (C#, VB, JavaScript и т. д.). XAML может использоваться как для браузер-базированных приложений, так и для настольных приложений.

XAML включает основные четыре категории элементов: панели, элементы управления, элементы, связанные с документом и графические фигуры. Заявлено 7 классов панелей, которые задают принципы отображения вложенных в них элементов. Для задания положения элементов относительно границ родительской панели используются атрибуты на манер свойств в объектно-ориентированных языках. Подобный синтаксис не совсем соответствует рекомендациям CSS, но является привычным для программистов настольных приложений.

Приложения, объявленные в XAML, могут включать множество страниц. Элемент управления PageViewer позволяет разбивать содержание на страницы и обеспечивает навигацию по ним. Элемент ContextMenu помогает в создании навигационных меню приложения. Код процедурного языка может быть размещён непосредственно в файле XAML или же назначен при сборке проекта.

В данном проекте я выбрал архитектуру MVVM.

Шаблон Model-View-ViewModel (MVVM) — применяется при проектировании архитектуры приложения. Первоначально был представлен сообществу Джоном Госсманом (John Gossman) в 2005 году как модификация шаблона Presentation Model. MVVM ориентирован на современные платформы разработки, такие как Windows Presentation Foundation, Silverlight от компании Microsoft, ZK framework.

MVVM используется для разделения модели и её представления, что необходимо для изменения их отдельно друг от друга. Например, разработчик задает логику работы с данными, а дизайнер соответственно работает с пользовательским интерфейсом.

Шаблон MVVM делится на три части:

1) Модель (англ. Model), так же, как в классической MVC, Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

2) Представление (англ. View) — это графический интерфейс, то есть окно, кнопки и т. п. Представление является подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновленное значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.

3) Модель Представления (англ. ViewModel) является, с одной стороны, абстракцией Представления, а с другой, предоставляет обёртку данных из Модели, которые подлежат связыванию. То есть, она содержит Модель, которая преобразована к Представлению, а также содержит в себе команды, которыми может пользоваться Представление, чтобы влиять на Модель.

3.2 Реализация СОД

СОД разрабатывалось на языке C# с использованием технологии uwp. Вёрстка реализовывалась на языке XAML в среде разработки Visual Studio 2017. Также использовался фреймворк Caliburn.Micro. Архитектура проекта – MVVM

Проект содержит 5 модулей. Их назначение описано в разделе 2.2 дипломного проекта. Рассмотрим каждый модуль по подробней.

Services модуль, где реализуются сервисы. Каталог services изображен на рисунке. 3.1.

Ключевым сервисом является QueryApiService. Данный сервис умеет делать универсальные post и get запросы. Все остальные api-сервисы пользуются его услугами.

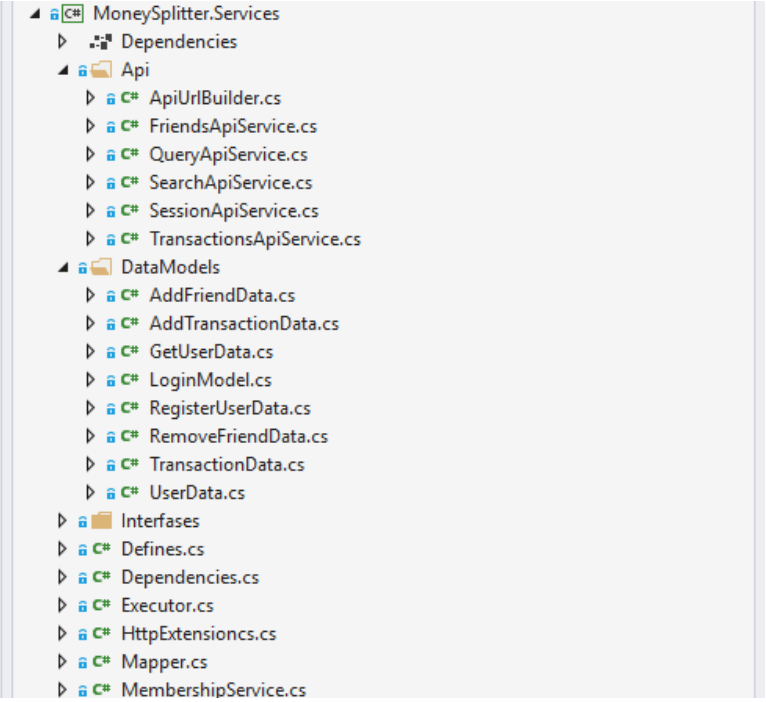


Рисунок 3.1– Содержание каталога Service

Ключевым сервисом является QueryApiService. Данный сервис умеет делать универсальные post и get запросы. Все остальные api-сервисы пользуются его услугами.

SearchApiService выполняет поисковые запросы. На сервер отправляет строку запроса, обратно получает найденных пользователей.

SessionApiServise отправляет запросы для регистрации и аутентификации пользователя.

FriendApiService отправляет запросы для различных манипуляций с друзьями пользователя (удаление, добавление, получение всех друзей).

TransactionApiService отправляет запросы для различных манипуляций с транзакциями (можно получить все транзакции, определенного друга). Все транзакции имеет участников. Участники делятся на три вида. Те, кто не отдавал еще деньги, те, которые отдали, но создатель транзакции еще не подтвердил это действие. И участники, которые завершили сделку. И сервис имеет возможность отправить запрос для изменения роли участника в данной транзакции.

ApiUrlBuilder строит url -ссылки для всех запросов к веб-серверу.

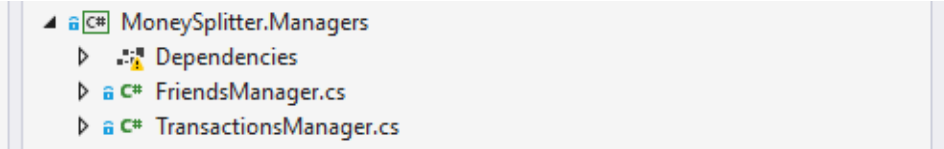
В каталоге DataModels хранятся модели для сохранения данных, которые приходят от веб-сервера. Сервер передают все данные в формате Jason и дынные потом конвертируются в наши модели. Эти модели нужны только для работы с веб сервером. В приложении они не используются, так как сервер иногда возвращает излишние данные.

Defines – статический класс для хранения всех констант.

Mapper класс для конвертации данных в сущность.

Executor – класс для обработки ошибок и сбоев.

Managers модуль, где реализуются менеджеры. Каталог изображён на рис. 3.2.

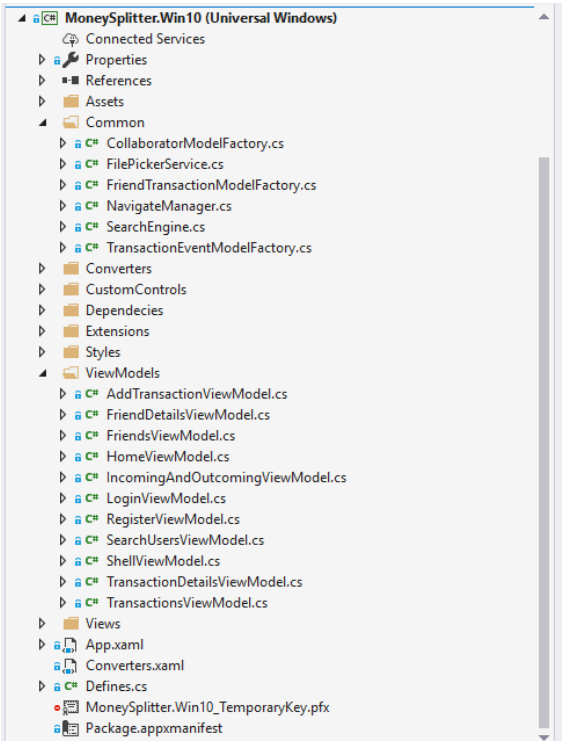


3.2 Содержание каталога Managers.

FrindsManger – использует FriendsApiService для различных функций (добавить, удалить или просмотреть всех друзей.

TransactionManger – использует TransactionApiService для управлениям транзакциями.

Win10 это модуль отвечающий за клиентскую часть приложения. Каталог изображён на рис. 3.3



3.3 Содержание каталога Win10.

Рассмотрим важнейшие составляющие проекта. ViewModels – это такие классы, которые имеют данные и действия, использующиеся в представлении. При изменении данных на вьюмодели, данные изменяются и в представлении. И если пользователь как-то взаимодействует с представлением меняет данные, они изменяются и во вьюмодели. Также, когда пользователь нажимает на кнопки, вызываются методы во вьюмодели.

В каталоге View находятся представлении. Это странички, написанные на языке XAML. Данные странички связанные с вьюмоделями. ShellView главная страница, располагается меню, и часть экрана которая изменяется. Другие представлении находятся в этой области экрана

3.3 Тестирование СОД

Условия тестирования: операционная система Windows 10 браузер. Для тестирования были сформированы данные и загружены на веб-сервер.

Приложения после установки появляется в меню пуска. Место нахождение изображено на рис 3.4.

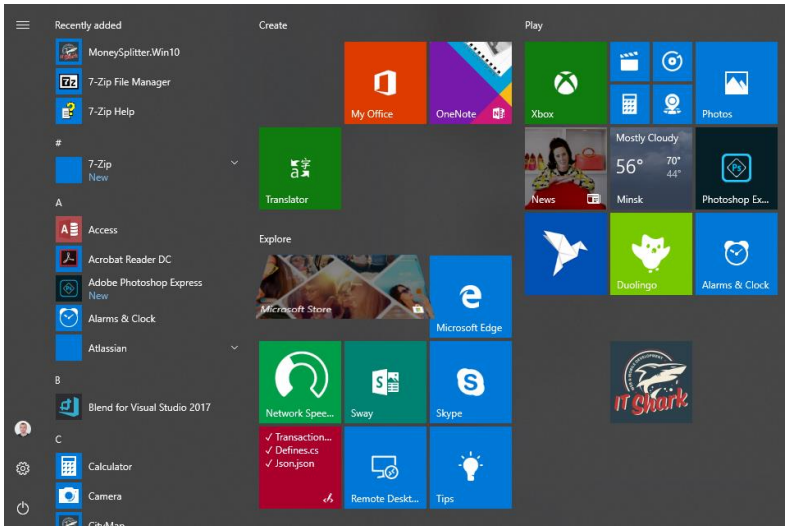


Рисунок 3.4 – Местонахождение приложения.

При открытии приложения должна открываться страница входа. На данной странице расположены две кнопки: «Вход» и «Регистрация». На которой присутствуют поля: «Email», «Пароль» и кнопка «Войти». Прототип этой страницы изображен на рисунке 3.5.

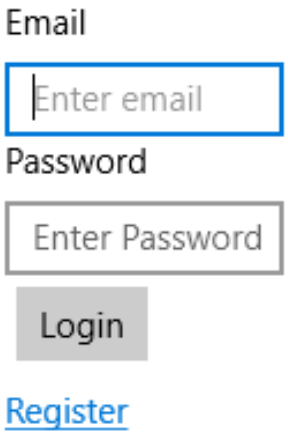


Рисунок – 3.5-Вход в систему

При нажатии на кнопку «Регистрация» страница входа должна заменятся на форму регистрации. На которой присутствуют поля: «Email», «Password», «Confirm password», «Name», «Surname», «Phone number», «Browse avatar image», «Browse background image» и кнопка «Registered». Это изображено на рисунке 3.6.

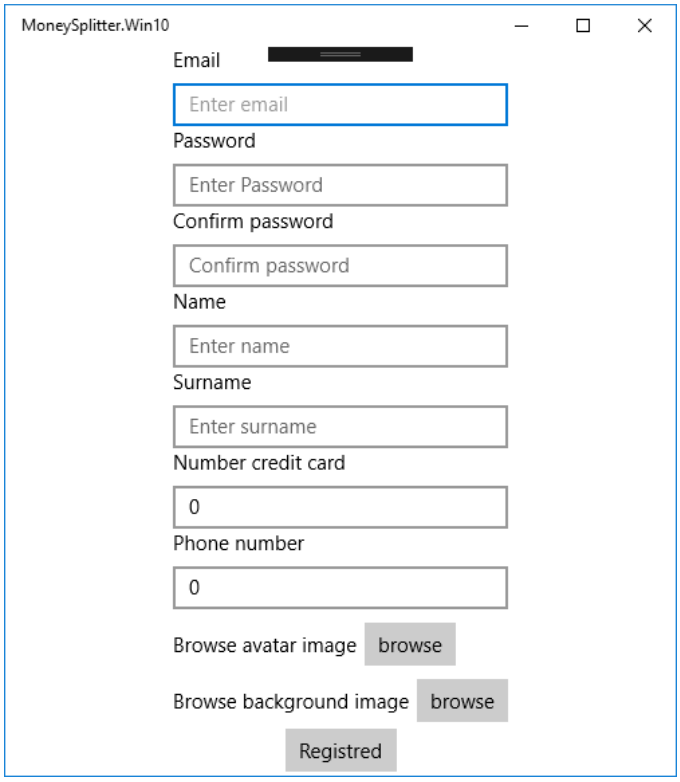


Рисунок 3.6 – Регистрация

Попытаемся зайти на домашнюю страницу Владислав Мудрый. Для этого введем логин и пароль. Это действие изображено на рис 3.7.

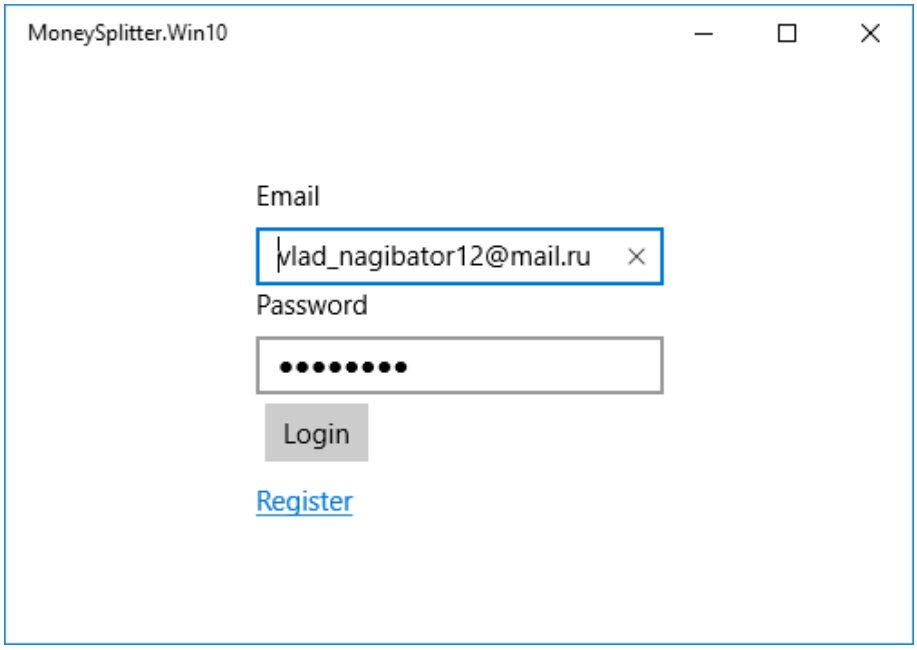


Рисунок 3.7 – Вход на страницу Владислава Мудрого

После входа в систему пользователь должен попасть домашнею страницу, где есть возможность просмотреть ближайшее событие, а также краткую сводку, сколько вам должны, вы должны. Домашняя страница пользователя изображена рисунке 3.8.

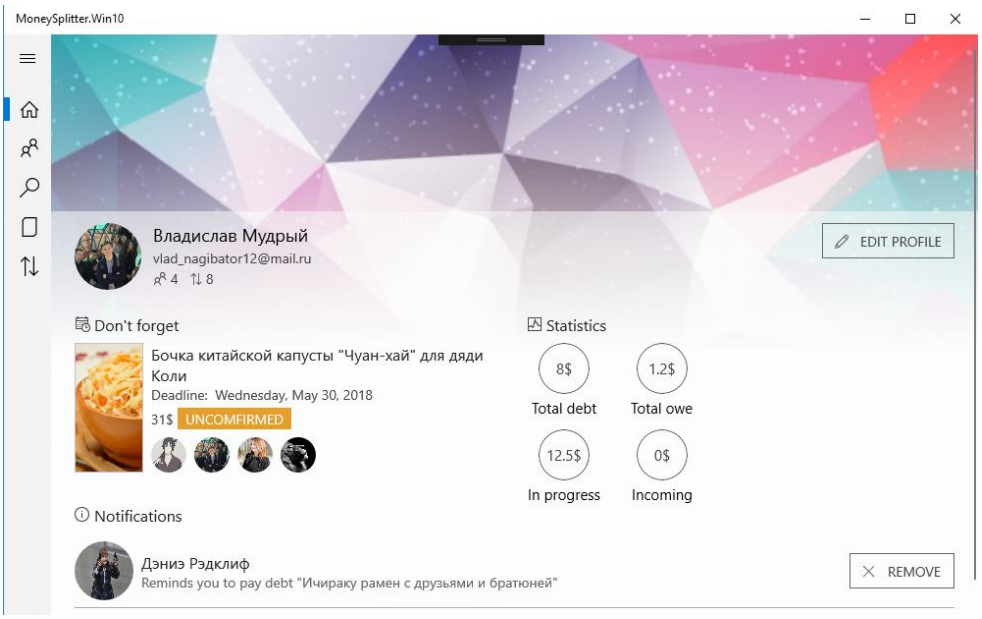


Рисунок 3.8 – Домашняя страница пользователя

Слева можно увидеть меню, которое имеет три режима, полностью свернутое (для мобильных телефонов), частично свернутое (для планшетов) и полностью развернутое (для компьютеров).

При выборе пункта «Friends» пользователь должен перейти на страницу, где представлены его друзья. Можно узнать подробную информацию друга, щелкнув на иконку друга. Также можно удалить друга, нажав на кнопку «Remove». Друзья пользователя изображены на рисунке 3.9.

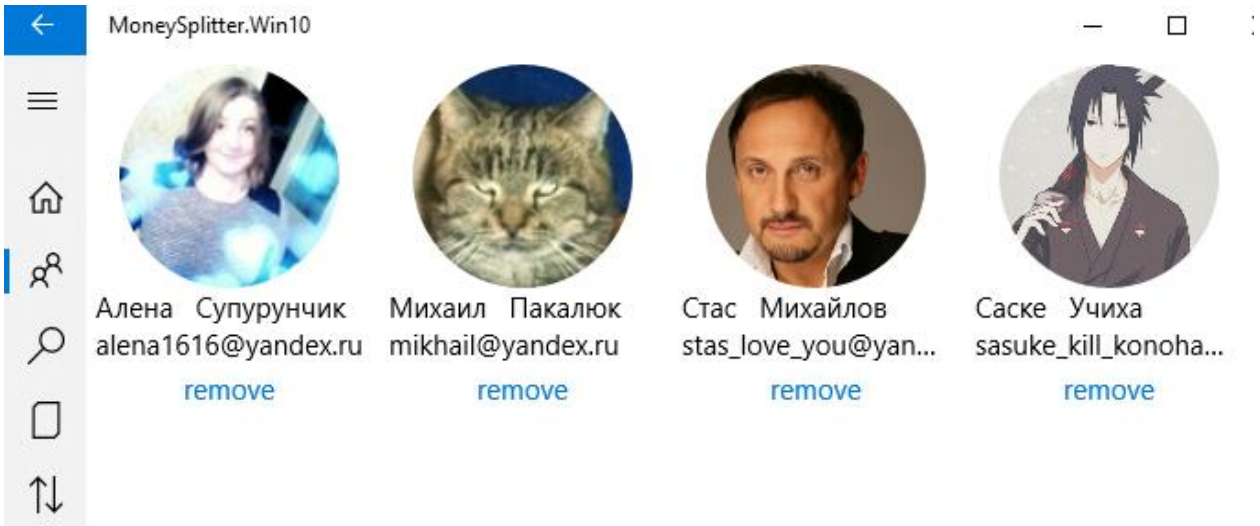


Рисунок 3.9 – Друзья пользователя

Попробуем удалить Михаила Пакалюка нажмем на кнопку «Remove». Результат удаления изображён на рисунке 3.10

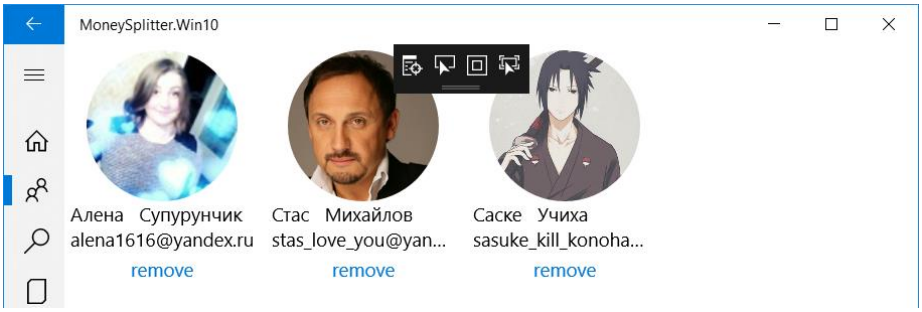


Рисунок 3.10 – Результат удаления

Нажмем на иконку Алена Супринчик мы должны перейти на страницу детали друга. Там мы увидит необходимую информацию, чтобы связаться с данным пользователем, это телефон и адрес электронной почты. А также имеется номер банковской карты, куда можно скинуть денежные средства. Справа подсчитана разница сумм всех долгов и займов, и можно увидеть сколько ты должен денег или сколько тебе должен друг. Пользователь имеет право погасить все сделки, если друг в жизни отдал эту сумму. Для этого нужно нажать на кнопку «Approve all».

Внизу страницы показаны следующие транзакции по отношению к пользователю, который зашел на данную страницу: долги друга, займы друга, транзакции в которых вы с другом участвуете и друг другу ничего не должны. Этим с данными транзакциями можно манипулировать используя кнопки справа от транзакций. Детали друга изображён на рисунке 3.11.

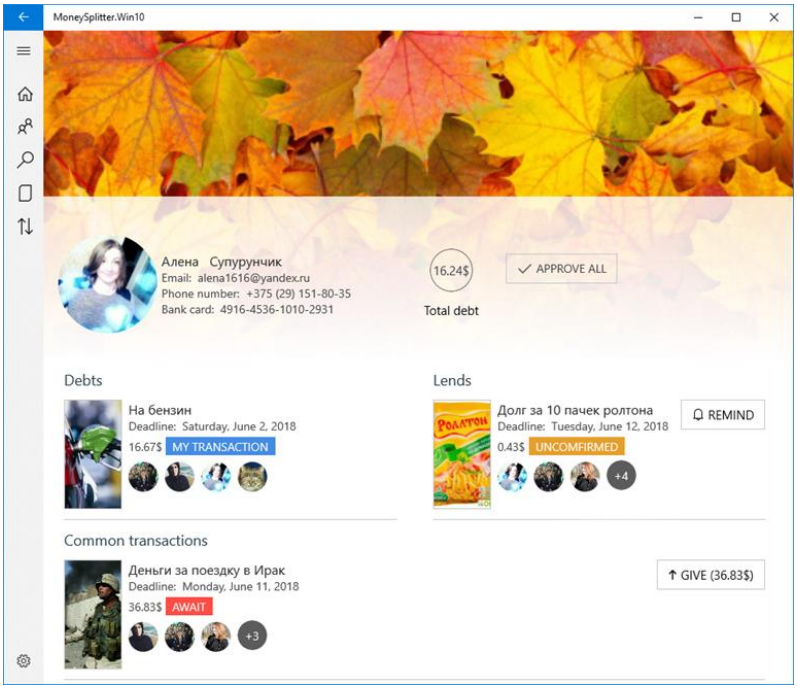


Рисунок 3.11– Детали друга

После нажатия на кнопку «Search» в меню, пользователь попадет непосредственно на страницу поиска пользователей. На странице будет располагаться форма для поискового запроса. Вводя строку, мгновенна будут отображаться найденные пользователи. Далее их можно добывать в друзья. Прототип этой страницы изображен на рисунке 3.12

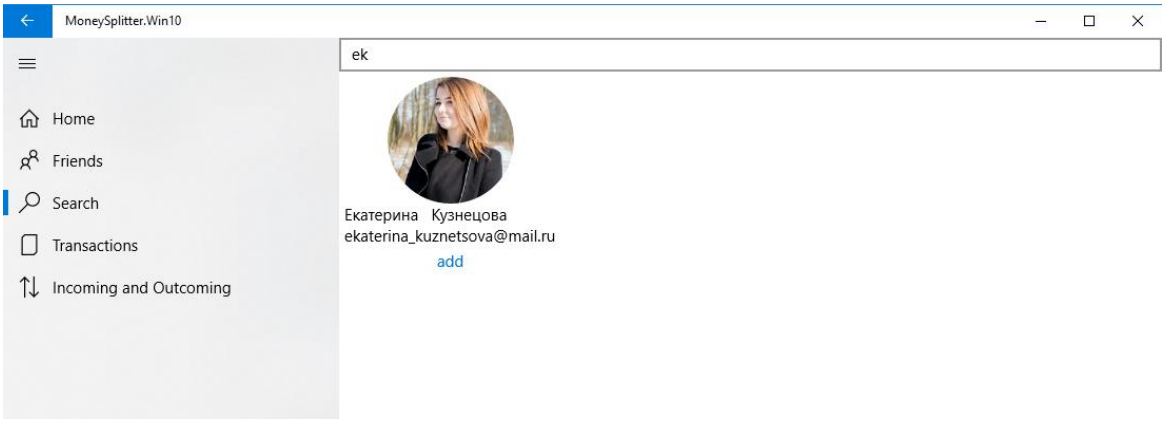


Рисунок 3.12 – Поиск пользователей

Попробуем Екатерину добавить друзья для этого нажмем на кнопку «add». А потом перейдем в друзья. Друзья пользователя изображены на рисунке 3.13

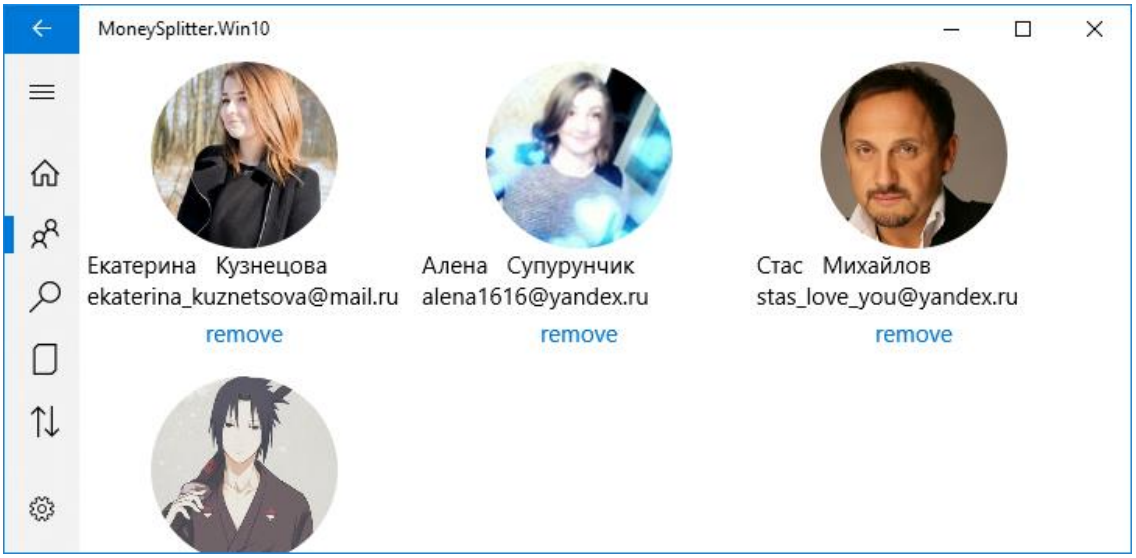


Рисунок 3.13 – Друзья пользователя

После нажатия на кнопку «Transactions» в меню, пользователь должен перейти на страницу транзакций пользователей. На странице будет располагаться список всех транзакций пользователя. Транзакции пользователя изображены 3.14. У каждой транзакция хранит следующую информацию. Название, дедлайн или дата события, стоимость, и метка определяющую роль пользователя в этой транзакции. А также некоторые транзакции имеет действии: отдать, напомнить. Вверху страницы есть кнопка

«Add», при нажатии должны перейти на страницу добавлении транзакции. Также есть кнопка для сортировки по категории, которая должна отсортировать по выбранной категории.

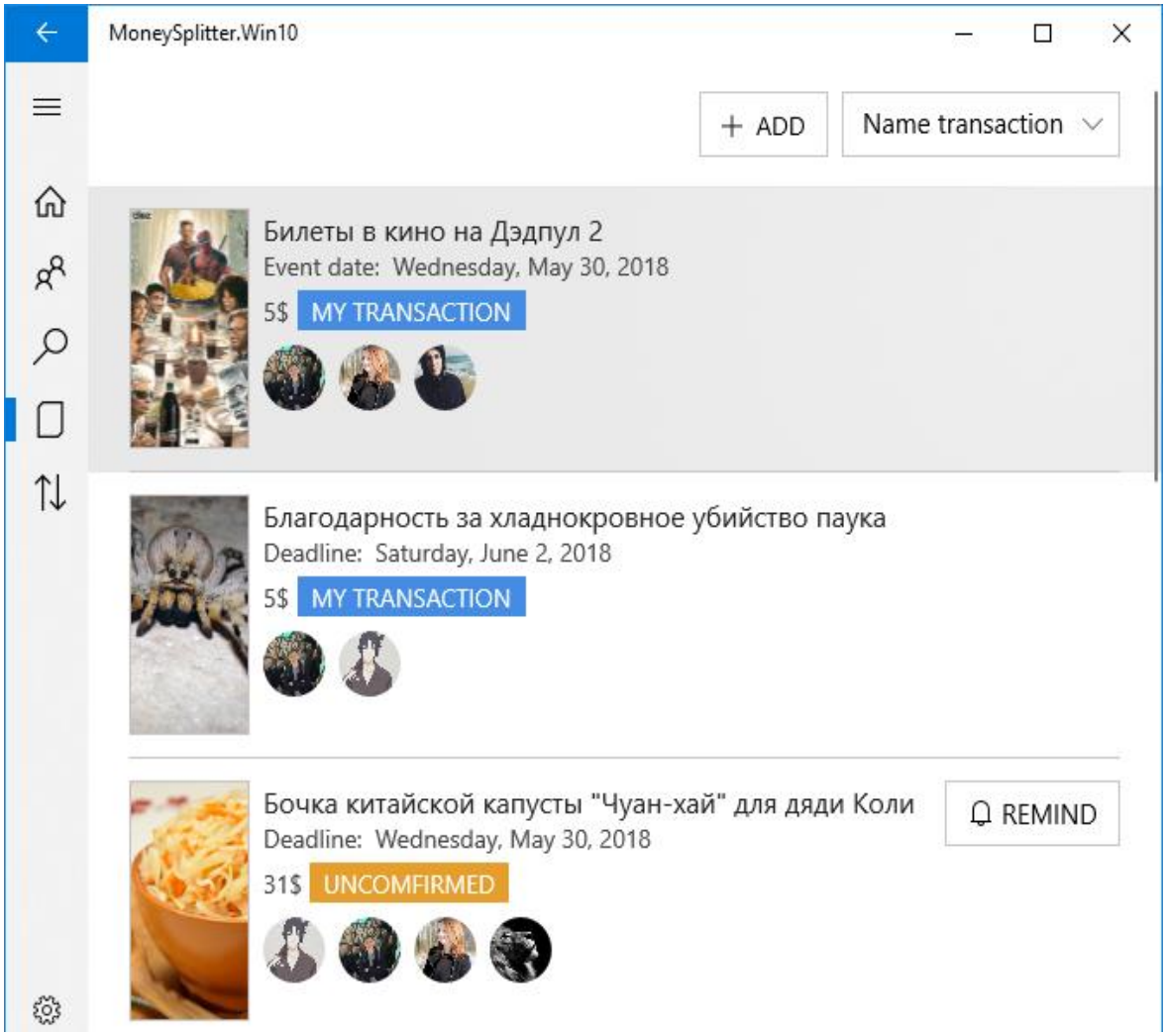


Рисунок 3.14 – Транзакции пользователя

При выборе транзакции должны открыться детали транзакции. Попробуем открыть транзакцию «Билеты в кино на Дэдпул 2». Да данной странице располагается подробная информация о транзакции. Небольшая информация об участниках транзакции, также имеется информация о владельце и прогрессе пользователей в данной сделке. Детали транзакции изображены на рисунке 3. 15.

При выборе пункта «Add» пользователь должен перейти на страницу для создание новой транзакции: имеются поля название, описание цена, дедлайн, изображение, и возможность выбора участников из друзей. Добавление транзакции изображено на рисунке 3. 16.

Давайте попробуем создать новую транзакцию. Пример создания изображен на рисунке 3.17.

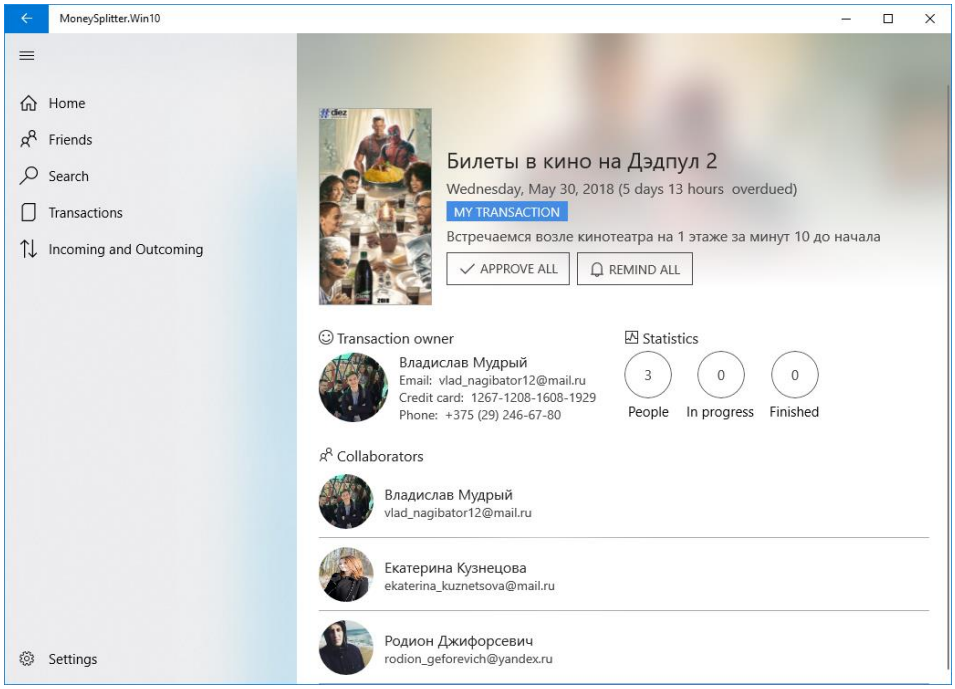


Рисунок 3.15 – Детали транзакции

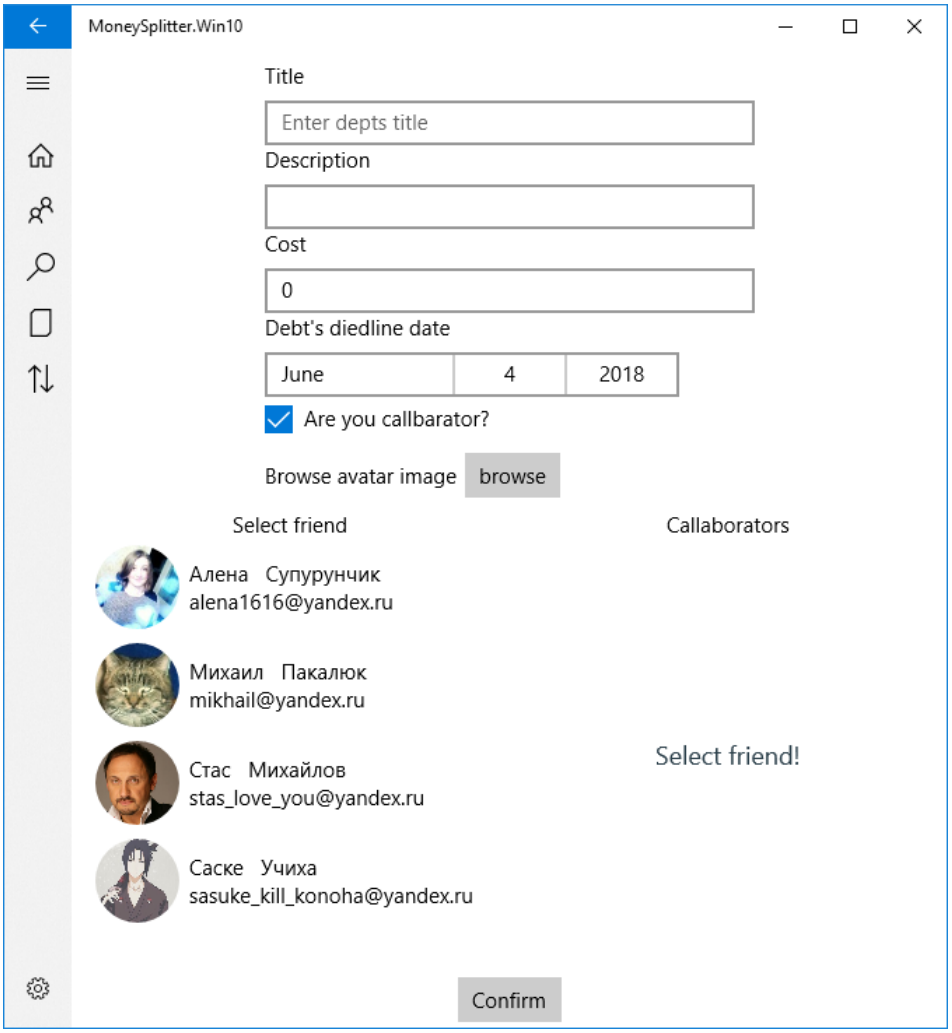


Рисунок 3.16 – Добавление транзакции

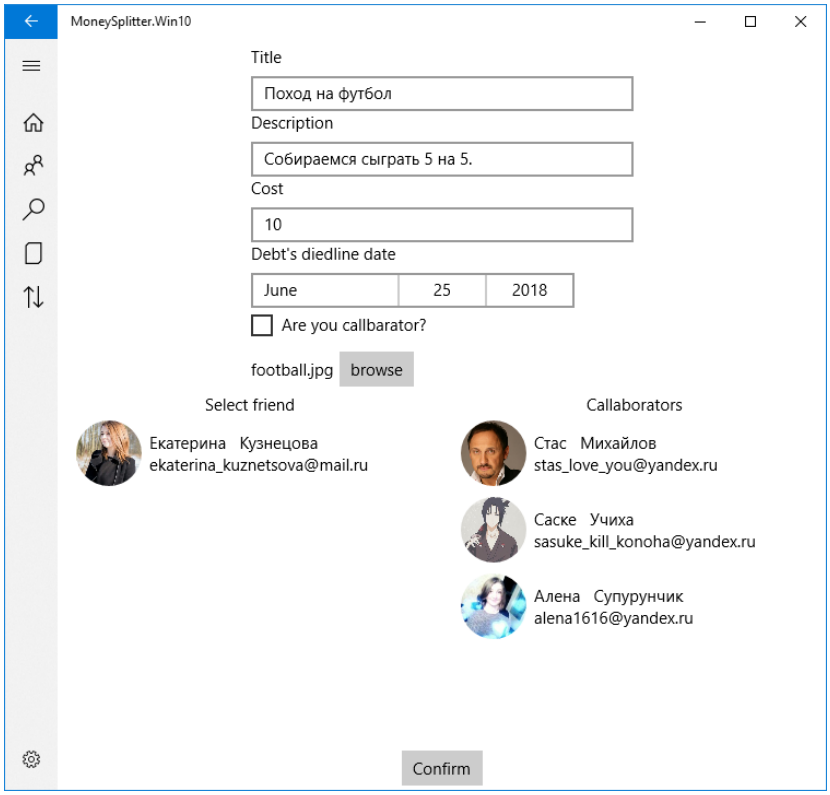


Рисунок 3.17 – Добавление транзакции «Футбол»

После нажатия на кнопку «confirm». Мы перейдем на страницу с транзакциями, где должна появится новая транзакция. Результат этого действия изображен на рисунке 3.18.

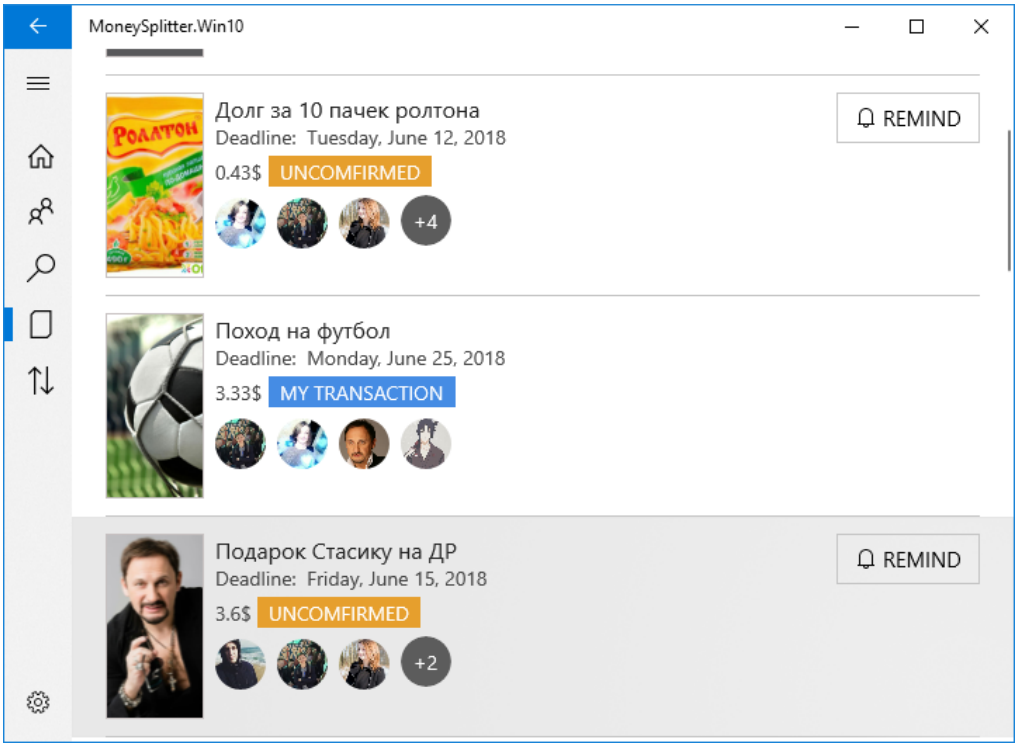


Рисунок 3.18 – Результат добавления транзакции

Сейчас попробуем отсортировать по цене. Для сортировки выберем параметр «cost». Результат сортировки изображён на рисунке 3.19

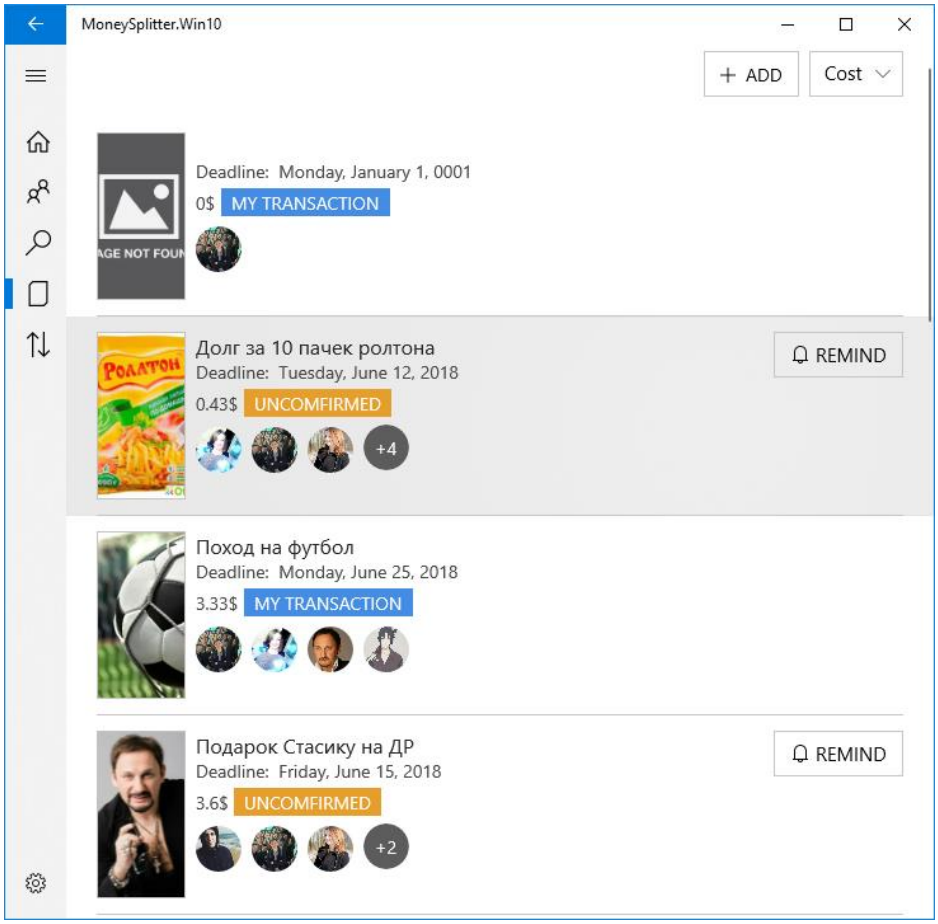


Рисунок 3.19 – Результат сортировки транзакций

После нажатия на кнопку «IncomingAndOutgoing» в меню, пользователь должен перейти на страницу с должниками и людьми, одолжившие пользователю. Показывает кто пользователю должен сколько, нажав на запись, происходит переход на детали пользователя. Должники и одолжившие изображены на рисунке 3. 20

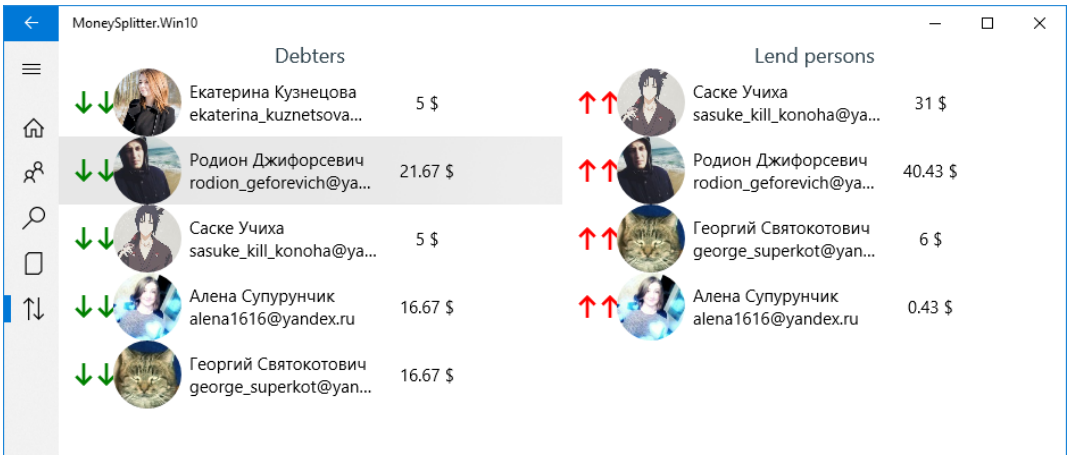


Рисунок 3.20 – Должники и одолжившие

Сейчас пробуем зарегистрировать пользователя. Рисунок 3.21. Должно зайти на домашнею страницу. Рисунок 3.22.

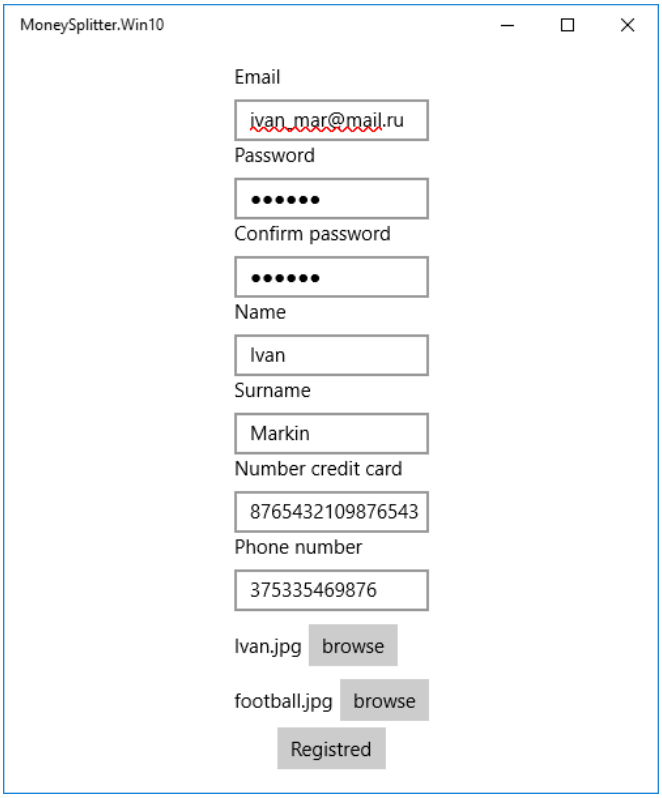


Рисунок 3.21 – Регистрация нового пользователя

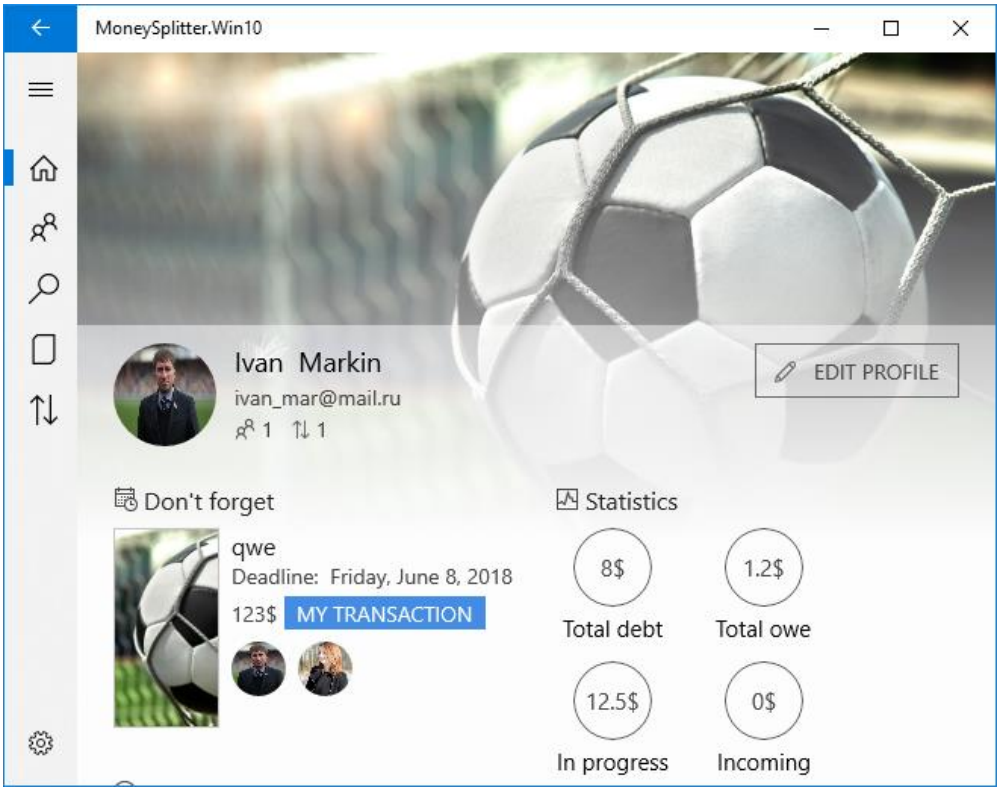


Рисунок 3.22 – Результат регистрации

Теперь попытаемся неправильно заполнить форму входа и регистрации и посмотрим на реакцию системы. Неправильный пароль при входе рисунок 3.23. Неправильный повторный пароль при регистрации рисунок 3.24.

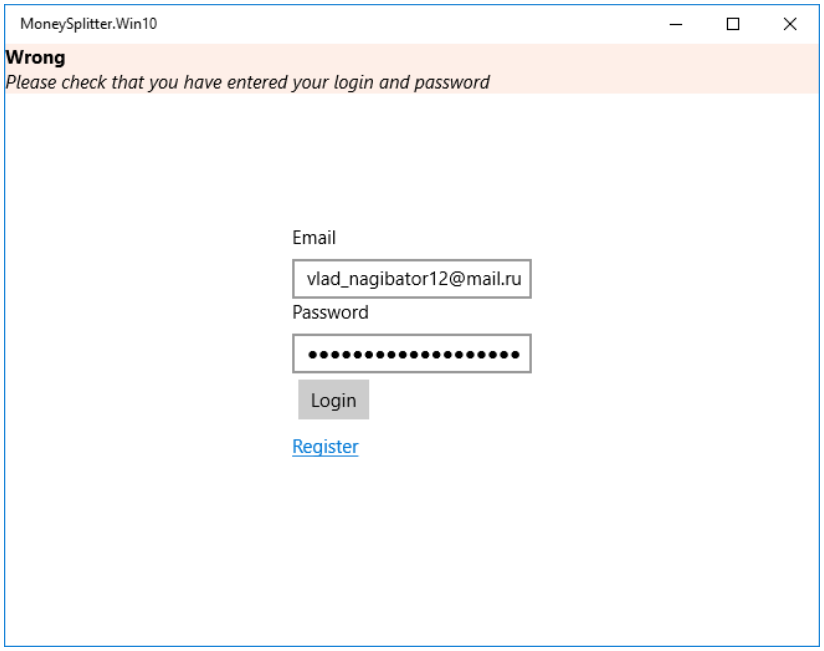


Рисунок 3.23 – Неправильный пароль при входе

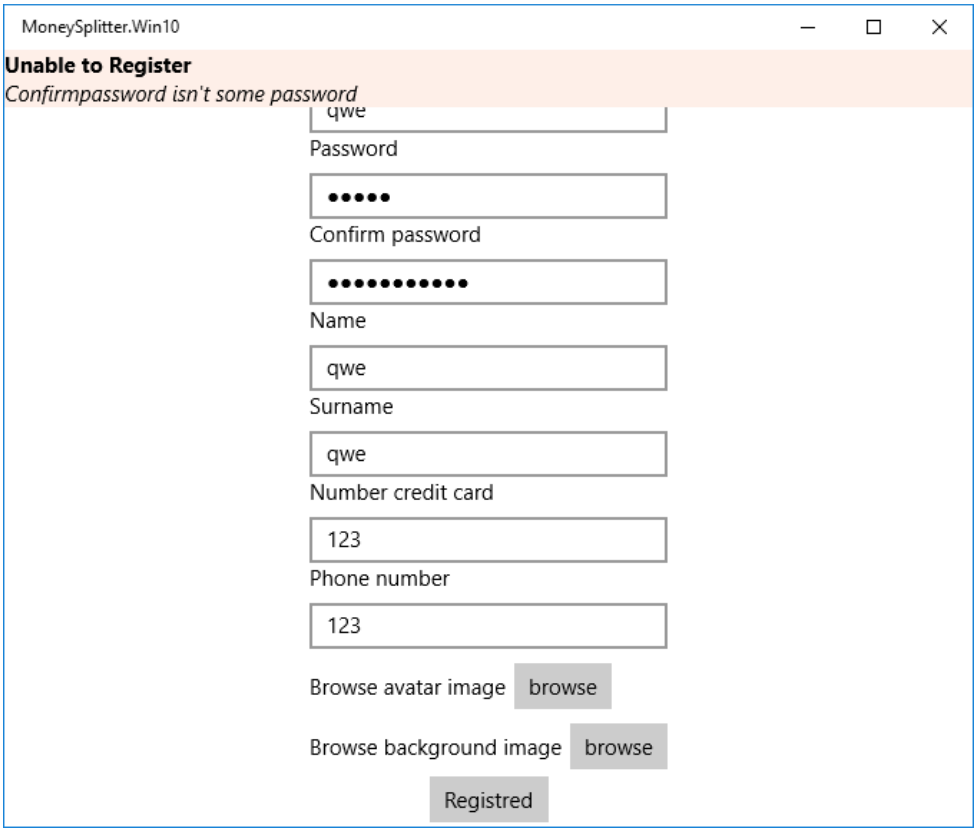


Рисунок 3.24 – Неправильный повторный пароль при регистрации

4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

4.1 Исходные данные для осуществления расчета

В данном дипломном проекте разрабатывается система для регулирования финансовых взаимоотношений группы людей. Данная система обеспечит сокращение затрат и времени на учет и обработку финансовых операций группы пользователей. Также она позволит хранить долги и займы пользователей. И пользователь непосредственно может воздействовать на должника, напомнить ему или подтвердит в приложении, то, что он отдал долг.

Разработка СОД предусматривает проведение всех стадий проектирования, относится ко второй группе сложности, т.к. разработка СОД предусматривает проведение практически всех стадий проектирования.

Последовательность расчетов:

1. Расчет объема функций программного модуля.
2. Расчет полной себестоимости программного продукта.
3. Расчет отпускной цены и чистой прибыли.

4.2 Расчет объема функций

Наименование проекта – «Разработка клиентского приложения MoneySpliter для перераспределения финансовых средств между пользователями».

Определение общего объема СОД.

Общий объем СОД (*Vo*) определяется исходя из количества и объема функций, реализуемых программой, по формуле (4.1):

$$V_o = \sum_{i=1}^{n} V_i \,, \qquad (4.1)$$

где *Vo* – общий объем СОД, *Vi* – объем отдельной функции СОД, *n* – общее число функций.

Расчет общего объема СОД (количества строк исходного кода) предполагает определение объема по каждой функции. Чаще всего на стадии технико-экономического обоснования проекта невозможно рассчитать точный объем функций, тогда данный объем может быть получен на основании ориентировочной оценки имеющихся

фактических данных по аналогичным проектам, выполненным ранее, или путем применения нормативов по каталогу функций.

Определение уточненного объема СОД.

На основании информации о функциях разрабатываемого СОД по каталогу функций определяется объем функций и общий объем СОД. В зависимости от организационных и технологических условий, в которых разрабатывается СОД, был скорректирован объем на основе экспертных оценок.

Среда разработки СОД – Visual Studio 2017

Уточненный объем СОД (*Vy*) определяется по формуле (4.2):

$$V_y = \sum_{i=1}^{n} V_{yi}, \qquad (4.2)$$

где *Vyi* – уточненный объем отдельной функции в строках исходного кода (см. таблицу 4.1).

Таблица 4.1 – Перечень и объем функций программного обеспечения

Код функции	Наименование (содержание) функции	Объем функции строк исходного кода (LOC)	
		По каталогу *Vi*	Уточненный *Vyi*
101	Организация ввода информации	130	110
102	Контроль, предварительная обработка и ввод информации	490	450
107	Организация ввода-вывода информации в интерактивном режиме	280	300
109	Управление вводом/выводом	1970	900
201	Генерация структуры базы данных	3500	1200
202	Формирование баз данных	1980	300
206	Манипулирование данными	7860	2400
207	Организация поиска и поиск в базе данных	4720	1290
506	Обработка ошибочных и сбойных ситуаций	1540	380
507	Обеспечение интерфейса между компонентами	1680	500
707	Графический вывод результатов	420	100
ИТОГО		24570	6930

На основании информации о функциях разрабатываемой системы обработки данных объемы некоторых функций были уменьшены и уточненный объем СОД (*Vy*) составил 6930 строки исходного кода (LOC) вместо 24570.

4.3 Расчет полной себестоимости ПО

Стоимостная оценка ПО у разработчика предполагает составление сметы затрат, которая включает следующие статьи расходов:

- заработную плату исполнителей (основную – *ЗПо* и дополнительную – *ЗПд*);
- отчисления на социальные нужды (*Рсоц*);
- материальные и комплектующие изделия (*Рм*);
- спецоборудование (*Рс*);
- машинное время (*Рмв*);
- расходы на научные командировки (*Рнк*);
- прочие прямые расходы (*Рпр*);
- накладные расходы (*Рнр*);
- затраты на освоение и сопровождение ПО (*Ро* и *Рсо*).

Полная себестоимость (ПО) разработки системы обработки данных (ПО) рассчитывается как сумма расходов по всем статьям с учетом рыночной стоимости и действующих нормативов организации-разработчика.

Основной статьей расходов на создание ПО является заработная плата проекта (основная и дополнительная) разработчиков (исполнителей) (ЗПо + ЗПд), в число которых принято включать инженеров-программистов, руководителей проекта. системных архитекторов, дизайнеров, разработчиков баз данных, веб-мастеров и других специалистов, необходимых для решения специальных задач в команде.

Расчет заработной платы разработчиков ПО начинается с определения:

- продолжительности времени разработки *Фрв*, которое устанавливается студентом экспертных путем с учетом сложности, новизны ПО и фактически затраченного времени. В данном дипломном проекте *Фрв* = 80 дней;
- количества разработчиков СОД. В данном дипломном проекте будет один разработчик – инженер программист 1 категории.

Заработная плата разработчика определяется как сумма основной и дополнительной заработной платы всех исполнителей.

Основная заработная плата исполнителя определяется по формуле:

$$ЗП_о = Т_{ст1\ р} * Т_к / 22 * Ф_{рв} * К_{пр}, \quad (4.3)$$

где Тст1 р - месячная тарифная ставка 1 разряда рабочего (с 1 марта 2018 года - 34 белорусских рубля);

$T_к$ - тарифный коэффициент согласно разряду исполнителя;

22 – среднее количество рабочих дней в месяце.

$Ф_{рв}$ - фонд рабочего времени исполнителя (продолжительность разработки ПО, дни);

$К_{пр}$ - коэффициент премий, $К_{пр}$ = 1,4.

Тарифный коэффициент инженера-программиста I категории согласно 13 разряду $T_к$ = 3,04. Продолжительность разработки ПО – 80 дня.

Дополнительная заработная плата исполнителя ($Н_{доп.зп}$) – 20%. Рассчитывается от основной заработной платы по формуле:

$$ЗП_д = ЗП_о * Н_{доп.зп} / 100, \quad (4.4)$$

Результаты вычислений внесем в таблицу 4.2.

Таблица 4.2 – Расчет заработной платы

Категории работников	Разряд	Тарифный коэффициент ($T_к$)	$Ф_{рв}$, дн.	Коэффициент премирования ($К_{пр}$)	$Н_{доп.зп}$, %	Заработная плата, бел. руб.		
						Основная	Дополнительная	Всего
Инженер-программист I категории	13	3,04	80	1,4	20	34 * 3,04 / 22 * 80 * 1,4 =526,20	526,20*20/100 = 105,24	526,20+ 105,24=631,44
Итого						526,20	105,24	631,44

Таким образом, как видно из таблицы 4.2, заработная плата инженера-программиста составляет 631,44 (бел. руб.).

Отчисления на социальные нужды ($P_{соц}$) определяются в соответствии с действующим законодательством по нормативу (34% - отчисления в ФСЗН + 0,6% отчисления по обязательному страхованию):

$$P_{соц} = (ЗП_{осн} + ЗП_{доп}) * \frac{34,6}{100}, \quad (4.5)$$

Поскольку приобретение спецоборудования не требуется, то данные расходы не будут рассчитываться.

Расходы по статье «Материалы» составляют 3% от $ЗП_о$.

Расходы по статье «Машинное время» ($P_{мв}$) включают оплату машинного времени, необходимого для разработки и отладки ПО. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени.

$$P_{мв} = Ц_{мі} * \frac{V_o}{100} * H_{мв}, \quad (4.6)$$

где $Ц_м$ – цена 1 машино-часа тыс. руб. (0,7 бел. руб);

V_o – уточненный общий объем функций строк исходного кода (LOC);

$H_{мв}$ – норматив расхода машинного времени на отладку 100 строк кода, машино-часов. Принимается в размере 0,8.

Поскольку научные командировки не предусмотрены, данная статья расходов не учитывается.

Расходы по статье «Прочие прямые затраты» ($P_{пр}$) включают затраты на приобретение специальной научно-технической информации и специальной литературы, рассчитываются по формуле (4.7). Определены в размере 10% от основной заработной платы исполнителей.

$$P_{пр} = ЗП_{осн} * \frac{H_{пр}}{100}, \quad (4.7)$$

где $H_{пр}$ – норматив прочих затрат.

Затраты по статье «Накладные расходы» ($P_{нр}$) связаны с содержанием вспомогательных хозяйств, а также с расходами на общехозяйственные нужды. Определяются по нормативу в процентах к основной заработной плате:

$$P_{нр} = ЗП_{осн} * \frac{H_{нр}}{100}, \quad (4.8)$$

где $H_{нр}$ – норматив накладных расходов, в %. В данном дипломном проекте норматив накладных расходов равен 50%.

Сумма выше перечисленных расходов по статьям на ПО служит исходной базой для расчёта затрат на освоение и сопровождение ПО:

$$Сумма\ \ затрат\ \ = ЗП_{осн} + ЗП_{доп} + Р_{соц} + Р_{м} + Р_{с} + Р_{мв} + Р_{нк} + Р_{пр} + Р_{нр}, \quad (4.9)$$

Затраты на освоение ПО (P_o) определяются по установленному нормативу от суммы затрат:

$$P_{o} = Сумма\ \ затрат\ \ * \frac{H_{o}}{100}, \quad (4.10)$$

где H_o – установленных норматив, %.

В данном дипломном проекте H_o – принимается равным 10%.

Затраты на сопровождение $P_{со}$ рассчитываются по формуле (4.11).

$$P_{со} = Сумма\ \ затрат\ \ * \frac{H_{со}}{100}, \quad (4.11)$$

где $H_{со}$ – норматив затрат на сопровождение.

Норматив затрат на сопровождение определен в размере 10%.

Полная себестоимость ($C_п$) разработки программного продукта рассчитывается как сумма расходов по всем статьям по формуле (4.12):

$$C_{п} = Сумма\ \ затрат\ \ + P_{o} + P_{со}, \quad (4.12)$$

Результаты вычислений внесем в таблицу 4.3.

Таблица 4.3 – Расчет себестоимости ПО

№ Пп	Наименование статей затрат	Норматив	Расчетная формула	Сумма затрат, руб.
1	2	3	4	5
1	Заработная плата, всего	-	-	631,44
1.1	Основная	-	-	526,20
1.2	Дополнительная	-	-	105,24
2	Отчисления на социальные нужды	34,6	631,44* 34,6 / 100	218,48
3	Спецоборудование	Не применялось		-
4	Материалы	3	631,44 * 3/100	18,94

Продолжение таблицы 4.3

№ Пп	Наименование статей затрат	Норматив	Расчетная формула	Сумма затрат, руб.
1	2	3	4	5
5	Машинное время	-	0,70* (6930 *0,8/ 100)	38,81
6	Научные командировки	Не планировались		-
7	Прочие прямые затраты	10	526,20* 10 / 100	52,62
8	Накладные расходы	50	526,20 * 50 / 100	263,1
9	Сумма затрат	-	631,44+ 218,48+18,94+ 38,81+ 52,62+263,1	1223,39
10	Затраты на освоение ПО	10	1223,39* 10 / 100	122,34
11	Затраты на сопровождение	10	1223,39* 10 / 100	122,34
12	Полная себестоимость	-	1223,39+ 122,34 + 122,34	1468,07

Полная себестоимость программного продукта составляет 1468,07 белорусских рублей.

4.4 Расчет отпускной цены и чистой прибыли

Для определения цены ПО необходимо рассчитать плановую прибыль.

Плановая прибыль рассчитывается по формуле (4.13):

$$П = C_п * \frac{R}{100}, \qquad (4.13)$$

где R – уровень рентабельности ПО.

В данном дипломном проекте уровень рентабельности принимается равным 30%.

После расчета прибыли от реализации определяется прогнозируемая цена ПО без налогов по формуле (4.14):

$$Ц_п = С_п + П, \quad (4.14)$$

где $С_п$ – полная себестоимость программного продукта, бел. руб;

$П$ – плановая прибыль от реализации ПО, бел. руб.

Отпускная цена (цена реализации) ПО включает налог на добавленную стоимость и рассчитывается по формуле (4.15):

$$Ц_о = С_п + П + НДС, \quad (4.15)$$

где $С_п$ – полная себестоимость программного продукта, бел. руб;

$П$ – плановая прибыль от реализации ПО, бел. руб.

$НДС$ – налог на добавленную стоимость, рассчитывается по формуле:

$$НДС = Ц_п * НДС / 100, \quad (4.16)$$

где $Ц_п$ – прогнозируемая цена, бел. руб;

$НДС$ – налог на добавленную стоимость, в настоящее время составляет 20%.

Прибыль от реализации ПО за вычетом налога на прибыль ($П_ч$) является чистой прибылью, остается организации разработчику и представляет собой экономический эффект от создания нового программного продукта:

$$П_ч = П * \left(1 - \frac{Н_п}{100}\right), \quad (4.17)$$

где $П$ – плановая прибыль от реализации ПО, бел. руб.

$Н_п$ – ставка налога на прибыль (в настоящее время 18%).

Все расчеты цены и прибыли по ПО сведены в таблицу 4.4.

Таблица 4.4 – Расчет цены и прибыли по ПО

№ пп	Наименование статей затрат	Норматив	Расчетная формула	Сумма затрат, руб.
1	Полная себестоимость	-	-	1468,07
2	Прибыль	30	1468,07 * 30 / 100	440,42
3	Прогнозируемая цена	-	1468,07+ 440,42	1908,49
4	НДС	20	1908,49* 20 / 100	381,70
5	Отпускная цена	-	1908,49 + 381,70	2290,18
6	Чистая прибыль	18	440,42* (1 - 18/100)	361,14

Итак, определены основные экономические показатели:

Полная себестоимость 1468,07 бел. руб.

Отпускная цена - 2290,18 бел. руб.

Чистая прибыль - 361,14 бел. руб.

Как видим из приведенных расчётов, данное ПО имеет выгоду и его разрабатывать целесообразно. Оно приносит разработчику чистую прибыль в размере 361,14 бел. руб. Можно сделать вывод о быстрой окупаемости продукта и его конкурентоспособности за счет современного пользовательского интерфейса и использования новейших технологий, т.е. разработка данного продукта приведёт к положительному экономическому эффекту.

5 ЭНЕРГО– И РЕСУРСОСБЕРЕЖЕНИЕ

5.1 Ресурсосбережение

Ресурсосбережение - это совокупность мер по бережливому и эффективному использованию фактов производства. Обеспечивается посредством использования ресурсосберегающих и энергосберегающих технологий, снижения фондоемкости и материалоемкости продукции, повышения производительности труда, сокращения затрат живого и овеществленного труда, повышения качества продукции, рационального применения труда менеджеров и маркетологов, использования выгод международного разделения труда и др. Способствует росту эффективности экономики, повышению ее конкурентоспособности.

В Республике Беларусь действует межгосударственный стандарт разработанный Межгосударственным Техническим комитетом по стандартизации МТК 111. Настоящий стандарт является основополагающим и устанавливает цель, задачи, объекты, основные принципы, термины и классификацию групп требований рационального использования и экономного расходования материальных ресурсов на всех стадиях жизненного цикла веществ, материалов, изделий, продукции при проведении работ и оказании услуг юридическим и физическим лицам. Целью стандартизации в области ресурсосбережения является создание организационно-методической и нормативной основы, необходимой и достаточной для проведения государственной технической политики, направленной на снижение ресурсоемкости получаемого дохода без ухудшения условий экономического развития страны при безусловном обеспечении высоких потребительских свойств продукции.

Требования ресурсосбережения подразделяют на три группы:

- требования ресурсосодержания, определяющие совершенство процессов, продукции, работ и услуг, например, по составу и количеству использованных материалов, массе, габаритам, объему изделия;
- требования ресурсоемкости (по технологичности), определяющие возможность достижения оптимальных затрат ресурсов при изготовлении, ремонте и утилизации продукции, а также выполнении различных работ и оказании услуг с учетом требований экологической безопасности;
- требования ресурсоэкономичности изделия, определяющие возможность достижения оптимальных затрат ресурсов при эксплуатации, ремонте и утилизации продукции, а также при выполнении работ и оказании услуг.

Указанные группы требований взаимосвязаны при:

- разработке продукции, планировании работ и услуг (устанавливают проектные требования ресурсосодержания и ресурсоэкономичности, рекомендации по ресурсоемкости);
- изготовлении продукции, выполнении работ и оказании услуг (устанавливают уточненные (контрольные) требования ресурсоемкости (по технологичности));
- эксплуатации продукции, выполнении работ и оказании услуг (устанавливают уточненные (контрольные) требования ресурсоэкономичности и ресурсоемкости));
- утилизации продукции (устанавливают требования ресурсоемкости и ресурсоэкономичности).

5.2 Энергосбережение

Энергосбережение (экономия энергии) — реализация правовых, организационных, научных, производственных, технических и экономических мер, направленных на эффективное (рациональное) использование (и экономное расходование) топливно-энергетических ресурсов и на вовлечение в хозяйственный оборот возобновляемых источников энергии. Энергосбережение — важная задача по сохранению природных ресурсов.

В целях укрепления экономической безопасности государства 14 июня 2007 года Президентом Республики Беларусь подписана Директива № 3 «Экономия и бережливость - главные факторы экономической безопасности государства».

Государственное регулирование в области энергосбережения и повышения энергетической эффективности осуществляется путем установления:

- требований к обороту отдельных товаров, функциональное назначение которых предполагает использование энергетических ресурсов;
- запретов или ограничений производства и оборота в Республике Беларусь товаров, имеющих низкую энергетическую эффективность, при условии наличия в обороте или введения в оборот аналогичных по цели использования товаров, имеющих высокую энергетическую эффективность, в количестве, удовлетворяющем спрос потребителей;
- обязанности по учету используемых энергетических ресурсов;
- требований энергетической эффективности зданий, строений, сооружений;
- обязанности проведения обязательного энергетического обследования;
- требований к энергетическому паспорту;

- обязанности проведения мероприятий по энергосбережению и повышению энергетической эффективности в отношении общего имущества собственников помещений в многоквартирном доме;
- требований энергетической эффективности товаров, работ, услуг, размещение заказов на которые осуществляется для государственных или муниципальных нужд;
- требований к региональным, муниципальным программам в области энергосбережения и повышения энергетической эффективности;
- требований к программам в области энергосбережения и повышения энергетической эффективности организаций с участием государства или городского образования и организаций, осуществляющих регулируемые виды деятельности;
- основ функционирования государственной информационной системы в области энергосбережения и повышения энергетической эффективности;
- обязанности распространения информации в области энергосбережения и повышения энергетической эффективности;
- обязанности реализации информационных программ и образовательных программ в области энергосбережения и повышения энергетической эффективности.

ЗАКЛЮЧЕНИЕ

В данном проекте представлены результаты создания модулей, которые помогают реализовывать планы и задачи.

В процессе дипломного проектирования выполнена следующая работа:

Изучен объект автоматизации и сформулирована постановка задачи на создание программной системы (приложение А).

Определена структура системы и спроектировано информационное и программное обеспечение, обозначена структура интерфейса для пользователей системы.

Выбраны инструментальные средства для реализации компонентов системы, реализована система и проведены испытания.

Разработан комплект документации для системы:

- описание применения (приложение Б);
- инструкция по установке и проверке (приложение В);
- текст программы (приложение Г).

Выполнено технико-экономическое обоснование, в рамках которого определена экономическая эффективность системы. Оно приносит разработчику чистую прибыль в размере 361,14 бел. руб.

Рассмотрены методы энерго и ресурсосбережения.

В настоящее время система находится на стадии опытной эксплуатации.

СПИСОК СОКРАЩЕНИЙ

БД – база данных
ОС – операционная система
СОД – средство обработки данных
ПО програмное обеспечение
UWP Universal Windows Plotform
XAML eXtensible Application Markup Language
MVVM Model View-ViewModel

СПИСОК ЛИТЕРАТУРЫ

1. ЕСПД. Техническое задание. ГОСТ 19.201-1978
2. ЕСПД. Описание применения. ГОСТ 19.502-1978
3. ЕСПД. Текст программы. ГОСТ 19.401-1978.
4. Кулакова Л.О., Кичаева Т.В. Методические указания по технико-экономическому обоснованию дипломных проектов для студентов специальности АСОИ. БрГТУ: Брест, 2015.
5. Стандарт БрГТУ по оформлению курсовых и дипломных работ
6. Stack Overflow – сайт вопросов и ответов для программистов. Режим доступа: https://stackoverflow.com, 20.05.2018
7. Habrahabr – русское сообщество программистов. Режим доступа: https://habrahabr.ru/ 25.05.2018
8. Официальная документация UWP –. Режим доступа: https://docs.microsoft.com/en-us/windows/uwp/index 25.05.2018