МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии и стандарты проектирования»

Тема «Семейный органайзер»

Исполнитель

студент 2 курса группы 4 Перкаль Анастасия Алексеевна

Руководитель работы ассистент Северинчик Н. А.

Курсовой проект защищён с оценкой

Председатель Пацей Н.В.

Минск 2021

Содержание

[Введение 3](#_Toc73125619)

[1 Аналитический обзор литературы 4](#_Toc73125620)

[2 Анализ требований к программному средству и разработка функциональных требований 7](#_Toc73125621)

[3 Проектирование программного средства 9](#_Toc73125622)

[3.1 Проектирование архитектуры приложения 9](#_Toc73125623)

[3.2 Проектирование базы данных 13](#_Toc73125624)

[4 Создание программного средства 16](#_Toc73125625)

[4.1 Реализация сущностей 16](#_Toc73125626)

[4.2 Реализация регистрации и аутенфикации 16](#_Toc73125627)

[4.3 Реализация представления 17](#_Toc73125628)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 19](#_Toc73125629)

[6 Руководство по использованию 24](#_Toc73125630)

[Заключение 34](#_Toc73125631)

[Список использованных источников 35](#_Toc73125632)

[Приложение А 36](#_Toc73125633)

[Приложение Б 37](#_Toc73125634)

[Приложение В 38](#_Toc73125635)

[Приложение Г 39](#_Toc73125636)

[Приложение Д 40](#_Toc73125637)

[Приложение Е 41](#_Toc73125638)

[Приложение Ж 42](#_Toc73125639)

[Приложение З 43](#_Toc73125640)

# Введение

Существует достаточное количество приложений для ведения учета семейного бюджета, десктопных ежедневников, планеров и других подобных программ, позволяющих перенести инструменты для контроля и организации жизни с бумаги на экран компьютера.

Курсовой проект посвящён разработке программного средства «Family Organizer» — десктопного помощника ведения семейного бюджета, просмотра статистики, заполнения списка покупок, а также составления планов.

При создании приложения преследовались две главные цели:

* простота и удобство в использовании как родителями, так и детьми;
* объединение вышеперечисленных инструментов организации жизни в одном месте.

«Family Organizer» помогает избавиться от бумажной работы, а также предоставляет инструменты для организации совместной работы членов семьи. Программа интуитивно понятна для пользователя, так как направлена на работу с задачами, с которыми сталкивается каждая семья.

«Family Organizer» имеет 2 типа пользователей: родитель(администратор) и ребенок. Родитель имеет доступ к скрытым для ребенка страницам приложения: статистике семейного бюджета и списку пользователей. Администратор также контролирует список покупок: принимает или отклоняет заявки детей.

В качестве интерфейса приклaдного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Хранение данных осуществляется в SQLite.

1 Аналитический обзор литературы

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

Теоретической основой для реализации курсового проекта была выбрана документация для разработчиков и технических специалистов с официального сайта Microsoft, доступная по ссылке <https://docs.microsoft.com/en-us/dotnet/csharp/>.

Также использовалась информация с сайта https://metanit.com/. Данный online-ресурс посвящён различным языкам и технологиям программирования, компьютерам, мобильным платформам и ИТ-технологиям. Здесь можно найти различные руководства и учебные материалы, статьи и примеры.

Следует отметить еще один сайт - http://professorweb.ru/. Сайт посвящен урокам по C# и платформе .NET Framework.

Статьи с https://habr.com/ru/ - уникальный источник реального опыта разработчиков и не менее опытных комментаторов, был также использован в ходе курсового проектирования.

Приложений с похожим функционалом на просторах Интернет достаточно. Например, в Microsoft Store иожно встретить такие приложения: «Budget Planning», «Mini Notes», «Список покупок!».

Краткий обзор упомянутых приложений представлен ниже.

Budget Planning — приложение, которое совмещает в себе функции планировщика бюджета и финансового калькулятора. Интерфейс «Budget Planning» представлен на рисунке 1.1.

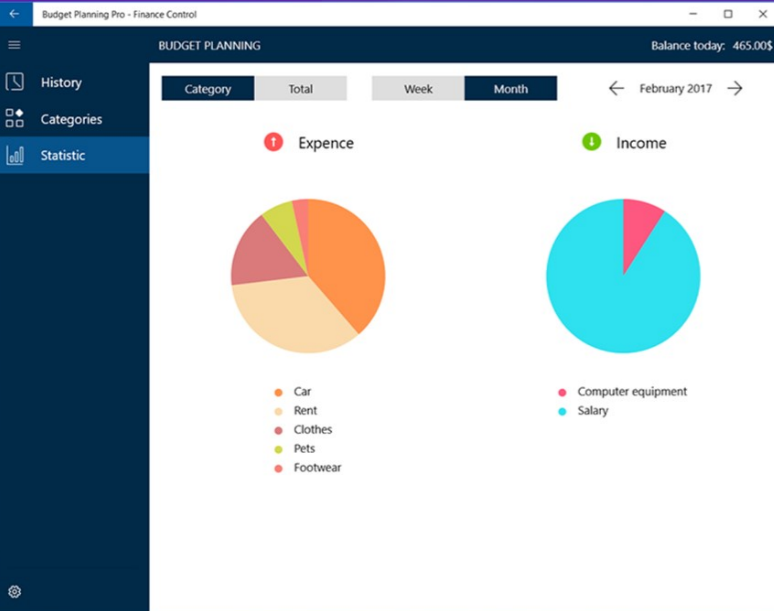


Рисунок 1.1 — Интерфейс «Budget Planning»

Проанализировав ПО «Budget Planning», можно выделить его основные минусы и плюсы.

Основные минусы:

* Диаграммы не интерактивные;
* Дизайн UI.

Основные плюсы:

* Просмотр статистики в виде круговой диаграммы;
* Визуальное отображение финансовой динамики.

В приложении «Family Organizer» для отображения графиков используется библиотека LiveCharts. Таким образом можно избежать минусов, описанных выше.

Список покупок! – приложение для создания списка покупок дома, которым можно воспользоваться в магазине. Интерфейс приложения представлен на рисунке 1.2.

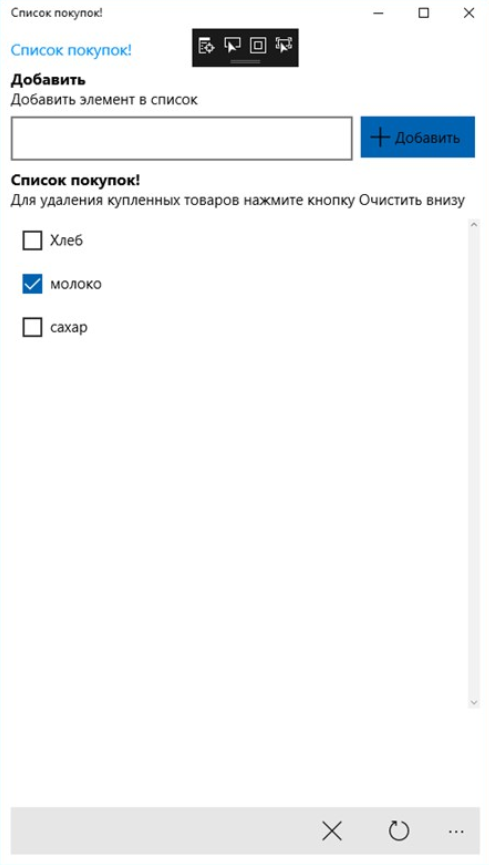


Рисунок 1.2 — Интерфейс «Список покупок!»

«Mini Notes» — приложение для менеджмента мини-записок.

К недостаткам приложения можно отнести дизайн UI. Основным достоинством «Mini Notes» является простота в использовании.

Интерфейс приложения представлен на рисунке 1.3.

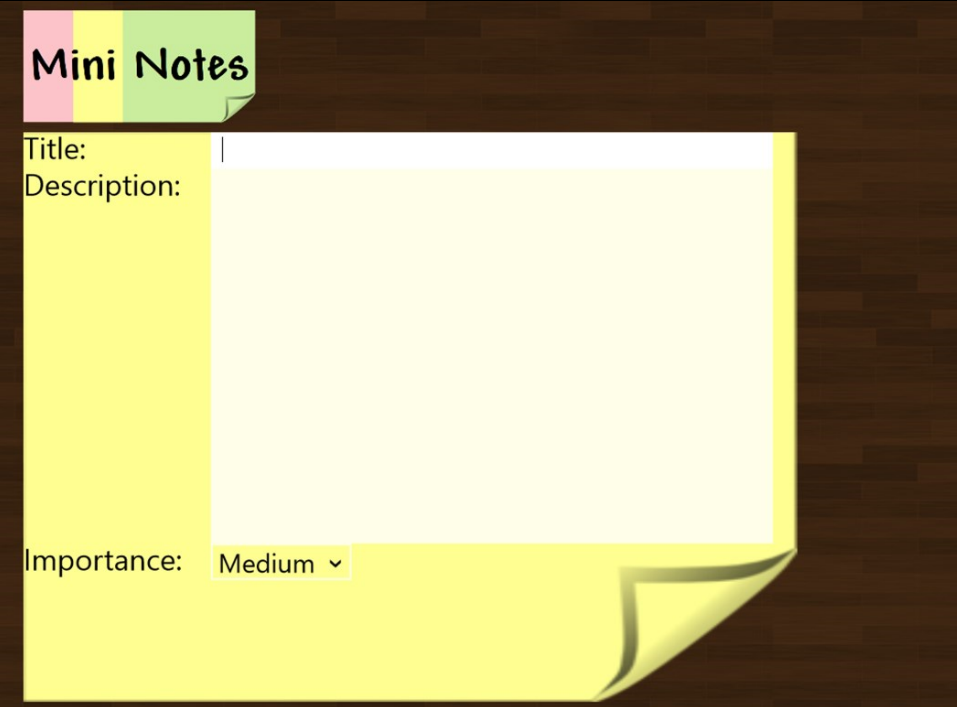


Рисунок 1.3 — Интерфейс «Mini Notes»

2 Анализ требований к программному средству и разработка функциональных требований

Программное средство должно быть реализовано на объектно-ориентированном языке программирования C# с использованием технологии WPF. Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Window Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует «урезанная» версия CLR, известная как WPF/E, она же известна как Silverlight (в проектировании не использовалась).

База данных была разработана в SQLite — компактной встраиваемой СУБД.

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы.

Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа.

Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

Для осуществления связи между базой данных и приложением на C# необходим «посредник», которым является технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Entity Framework позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Программное средство предоставляет пользователю функциональные возможности, которые можно наглядно продемонстрировать на UML-схеме, расположенной в приложении А.

После проведения анализа были выявлены функциональные требования, описанные ниже. Приложение должно:

* хранить всю информацию в базе данных;
* производить валидацию вводимых пользователем данных;
* корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* предоставлять возможность просмотра статистики бюджета по заданному месяцу;
* предоставлять возможность составления списка покупок, планов на день;
* предоставлять возможность оставлять краткие комментарии на главной странице;
* предоставлять возможность редактирования профиля;
* предоставлять инструменты для контроля семейного бюджета.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

3 Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

# 3.1 Проектирование архитектуры приложения

Архитектура, управляемая событиями (Event-driven architecture), является шаблоном архитектуры программного обеспечения, позволяющим реализовать создание, определение, потребление и реакцию на события.

Событие можно определить как «существенное изменение состояния».

Этот архитектурный шаблон может применяться при разработке и реализации приложений и систем, передающих события среди слабосвязанных программных компонентов и служб.

Программными компонентами в данном случае являются разметка с ЭУ, отправляющими события, и обработчики событий.

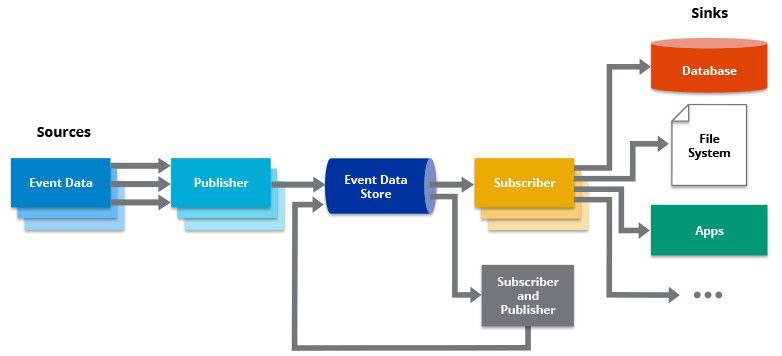


Рисунок 3.1 — Схема event-driven architecture

Система, управляемая событиями, обычно содержит источники событий (или агентов) и потребителей событий (или стоков). Стоки ответственны за ответную реакцию, как только событие возникло. Реакция может полностью или не полностью создаваться стоком. К примеру, сток может отвечать лишь за фильтрацию, преобразование и доставку события другому компоненту, либо он может создать собственную реакцию на это событие. Первая категория стоков может основываться на традиционных компонентах, таких как промежуточное программное обеспечение для обмена сообщениями, а вторая категория стоков (формирующая собственную реакцию в процессе работы) может потребовать наличия более подходящей платформы выполнения транзакций.

Создание приложений и систем в рамках архитектуры, управляемой событиями, позволяет им быть сконструированными способом, способствующим лучшей интерактивности, поскольку системы, управляемые событиями, по структуре более ориентированы на непредсказуемые и асинхронные окружения.

На рисунке 3.2 представлен принцип работы приложения с точки зрения родителя(администратора).

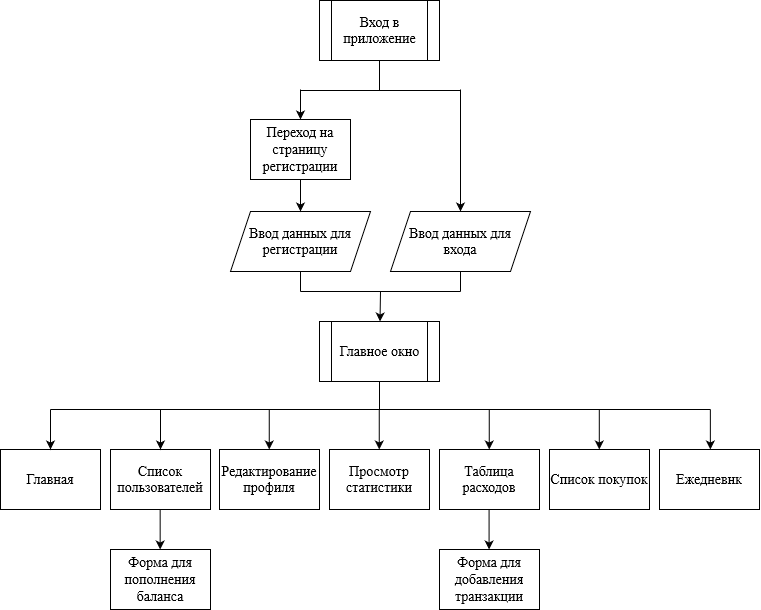


Рисунок 3.2 — Схема навигации по страницам для пользователя с ролью «Родитель»

На диаграмме 3.2 видно, что администратор при запуске приложения будет попадать на страницу входа, на которой чтобы перейти дальше необходимо ввести соответствующие данные или перейти на страницу регистрации и ввести данные для регистрации.

После входа в приложение пользователю откроется окно приложения с 7 вкладками:

* Главная;
* Список пользователей;
* Редактирование профиля;
* Просмотр статистики;
* Таблица расходов;
* Список покупок;
* Ежедневник.

На странице «Главная» будет отображаться краткая информация о пользователе: имя пользователя, фотография, роль и баланс. На этой странице пользователь сможет написать короткое сообщение, которое будет видно всем пользователям.

В разделе «Список пользователей» отображаются карточки со всеми пользователями, кроме текущего. В этом разделе администратор может регистрировать и удалять пользователей. При нажатии на кнопку «Добавить» открывается страница «Форма для пополнения баланса», где можно пополнить баланс пользователя.

На странице «Редактирование профиля» можно выбрать фотографию профиля, изменить логин и пароль.

На странице «Таблица расходов» размещена таблица со столбцами: дата, сумма транзакции, тип транзакции и имя пользователя. При нажатии на кнопку «Добавить» открывается страница «Форма для добавления транзакции», диаграмма деятельности которой расположена в приложении Б.

В разделе «Просмотр статистики» располагается график, иллюстрирующий баланс семьи в течение месяца, а также круговая диаграмма, в которой по категориям разбиты расходы в течение месяца.

На странице «Список покупок» размещен сам список покупок, в который можно добавлять и удалять записи, а также список заявок от пользователей с ролью «Ребенок». Администратор может либо одобрить, либо отклонить заявку.

В разделе «Ежедневник» размещен сам список дел, а также список рутинных дел, которые после добавления можно добавить в основной список, нажав на кнопку добавить, и удалить из списка дел без удаления из списка рутинных дел.

При авторизации пользователя с ролью «Ребенок», страницы «Список пользователей» и «Просмотр статистики» отсутствуют.

На странице «Список покупок» пользователь не может редактировать имеющийся список, а может только оставлять заявку.

На рисунке 3.3 представлен принцип работы приложения с точки зрения ребенка.

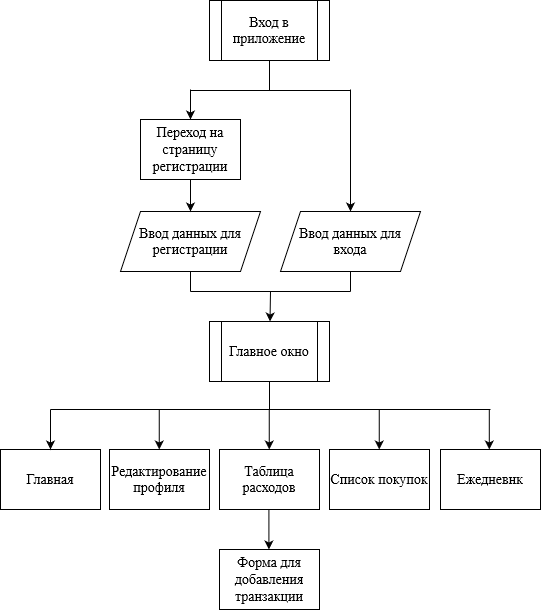


Рисунок 3.3 — Схема навигации по страницам для пользователя с ролью «Ребенок»

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД SQLite. В приложении В расположена диаграмма компонентов.

# 3.2 Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Проектирование базы данных проводится в два этапа: концептуальное и логическое проектирование.

Концептуальное проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня aбстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности − это любая детaль, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 7 сущностей:

* пользователи;
* балансы;
* комментарии;
* фотографии;
* покупки;
* дела;
* транзакции.

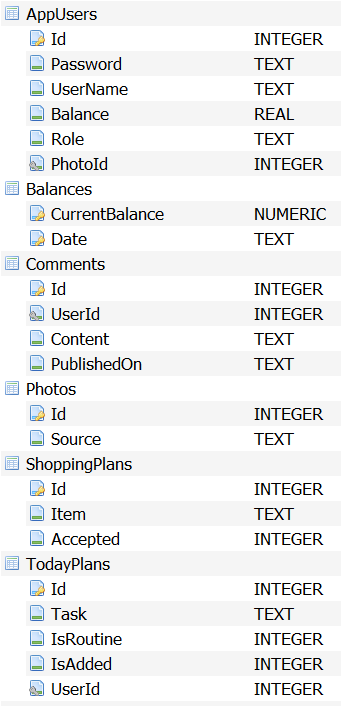
Также в ER-модели были определены необходимые связи. Например, между сущностями пользователь и транзакции была установлена связь один-ко-многим.

Логическое проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели дaнных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Диаграмма классов представлена в приложении Г.

Всего в базе данных содержится 7 таблиц. В таблице AppUsers хранятся все пользователи, зарегистрированные в приложении. Таблица Balances содержит информацию о суммарном балансе всех пользователей за определенную дату. Таблица Comments хранит оставленные пользователями комментарии на главной странице. Таблица Photos содержит фотографии, из которых пользователь выбирает себе аватар. Таблица ShoppingPlans хранит информацию о добавленном элементе, а также о его статусе (одобрено/не одобрено). Таблица TodayPlans хранит планы и их тип(рутина/не рутина). Таблица Transactions содержит информацию о депозитах, переводах денег между пользователями, а также о расходах.

Информация о таблицах и их столбцах представлена на рисунке 3.4.



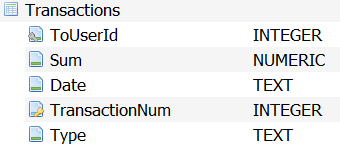


Рисунок 3.3 — Таблицы базы данных приложения «Family Organizer»

## **3.4 Проектирование логики сценариев использования**

При проектировании приложения были рассмотрены различные сценарии использования. В приложении Д представлена диаграмма последовательности добавления пользователя.

4 Создание программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

4.1 Реализация сущностей

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Database-First. Созданные модели объектов совпадают с сущностями, которые были сформированы раннее в разделе 3.2.

Диаграмма классов UML для сущностных классов представлена в приложении Г.

На диаграмме классов видно, как связаны между собой сущности. Сущность AppUser связана с классами TodayPlan, Comment, Transaction отношением композиции т.е. при уничтожении объекта AppUser в области памяти вместе с ним будут уничтожены и объекты TodayPlan, Comment, Transaction связанные с ним. И в этом плане объект User является главным, а объекты TodayPlan, Comment, Transaction – зависимыми.

Из диаграммы видно, что класс AppUser и класс Photo связаны между собой отношением агрегации. При агрегации реализуется слабая связь, то есть в данном случае объекты AppUser и Photo будут равноправны.

Связи между сущностными классами были настроены с помощью соглашений, а также Fluent API. Листинг настройки класса контекста с помощью Fluent API приведен в приложении Е.

4.2 Реализация регистрации и аутенфикации

Поскольку база данных SQLite хранится в памяти, опытный пользователь при желании может просмотреть ее и, например, узнать пароль администратора.

Для предотвращения подобных случаев в программном средстве реализовано хэширование паролей с применением стандарта PBKDF2.

PBKDF2 (Password-Based Key Derivation Function) — стандарт формирования ключа на основе пароля.

Соль (также модификатор входа хэш-функции) — строка данных, которая передаётся хеш-функции вместе с входным массивом данных (прообразом) для вычисления хэша (образа).

При регистрации, в случае успешного прохождения валидации пароля, он хэшируется со случайно сгенерированной солью (salt) и сохраняется в базу данных в формате xxxx:salt:hash, где

хххх – количество циклов хэширования;

hash – хэшированный пароль;

salt – соль.

Код, осуществляющий хэширование приведен в приложении Ж.

На рисунке 4.1 изображен пример хранения хэшированного пароля, хранящегося в базе данных.

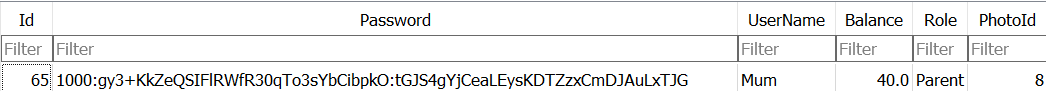


Рисунок 4.1 — Строка таблицы «AppUsers»

4.3 Реализация представления

Для разработки графической части приложения была выбрана технология WPF.

Windows Presentation Foundation (WPF) — это библиотека для создания пользовательских интерфейсов для интеллектуальных клиентских приложений. В основе WPF лежит мощная инфраструктура, основанная на DirectX — API-интерфейсе графики с аппаратным ускорением, который обычно используется в современных компьютерных играх. Это означает, что применение развитых графических эффектов не приведёт к снижению производительности.

Одной из важных особенностей WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Разработка с использованием XAML позволяет отделить графический интерфейс от логики приложения, а также создавать насыщенный интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#.

В таблице 4.1 приведено описание окон и страниц приложения «Family Organizer».

Таблица 4.1 — Описание окон и страниц приложения «Family Organizer»

|  |  |
| --- | --- |
| **Название** | **Краткое описание** |
| AddMoneyAdmin.xaml (окно) | Окно для пополнения баланса пользователя |
| AddTransaction.xaml (окно) | Окно для добавления транзакции |
| Diary.xaml (страница) | Страница для ведения списка дел |
| Edit.xaml (страница) | Страница для редактирования профиля |
| Family.xaml (страница) | Страница для удаления/добавления пользователей |
| FamilyOrganizerMessageBox.xaml (окно) | Окно для сообщений |
| General.xaml (страница) | Главная страница |
| Login.xaml (окно) | Окно для входа в приложение |
| MainWindow.xaml (окно) | Главное окно приложения |
| Money.xaml (страница) | Страница для просмотра списка транзакций |
| Register.xaml (окно) | Окно регистрации нового пользователя |
| ShoppingList.xaml (страница) | Страница для заполнения списка покупок |
| SplashScreen.xaml (окно) | Окно, выполняющее функцию загрузочного экрана |
| Statistics.xaml (страница) | Страница для просмотра статистики |

Логика переключения страниц отображена в приложении З.

5 Тестирование, проверка работоспособности и анализ полученных результатов

Прежде всего были проведены тесты авторизации: проверка на пустые и неверные данные. До тех пор, пока введённые данные не будут корректными, будет выводиться сообщение с пояснениями и переход на главную страницу не будет выполнен. Сообщения об ошибках представлены на рисунках 5.1 и 5.2.

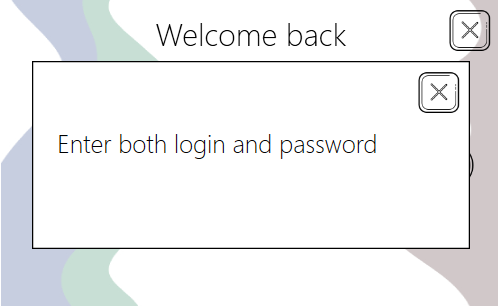


Рисунок 5.1 — Сообщение об ошибке

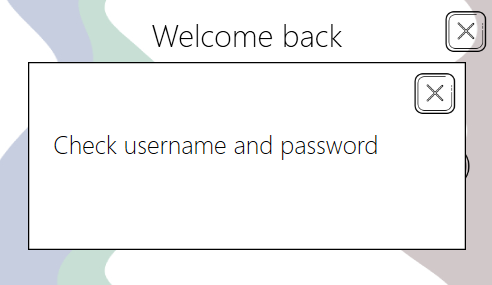


Рисунок 5.2 — Сообщение об ошибке

Валидация на странице регистрации не допускает ввода некорректных значений в поля для ввода логина и пароля. Оба поля являются обязательными. Пароль должен состоять минимум из 6 символов. Имя пользователя может состоять максимум из 10 символов, а пароль — из 30.

Примеры ошибок, которые могут возникнуть на этапе регистрации, представлены на рисунках 5.3 и 5.4.

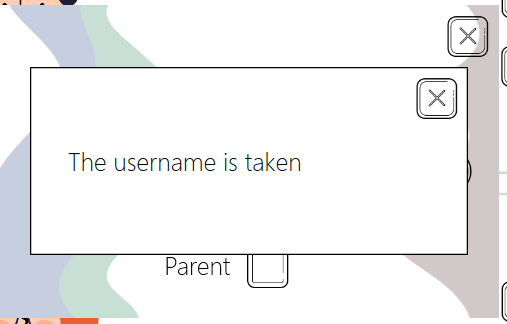


Рисунок 5.3 — Сообщение об ошибке

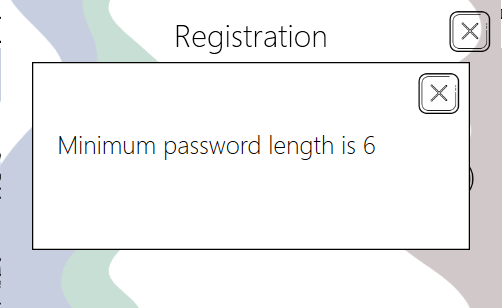


Рисунок 5.4 — Сообщение об ошибке

Также были проведены тесты на работу валидации внутри приложения. Наличие валидации является обязательным в связи с тем, что в приложении постоянно ведётся работа с базой данных.

Валидация организованна путем использования регулярных выражений, сообщений об ошибке, а также недоступности некоторых ЭУ.

Пример результата валидации данных и обработки различных экстремальных ситуаций внутри приложения приведены на рисунках 5.5 — 5.10.

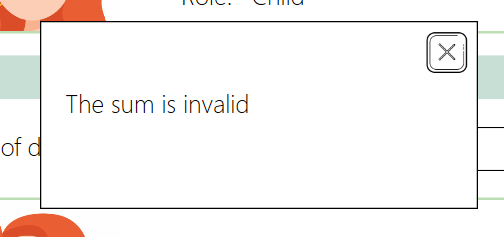


Рисунок 5.5 — Попытка ввести в поле для ввода суммы невалидные данные

При выборе в конструкторе типа транзакции «Expense», будет доступен только выбор категории и поле ввода суммы.

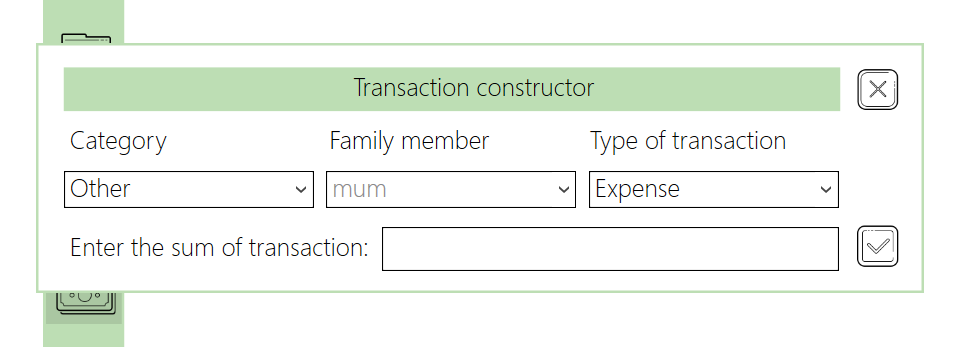


Рисунок 5.6 — Тип транзакции «Expense»

Если пользователь выбирает тип «Deposit», то будет доступно только поле ввода суммы.

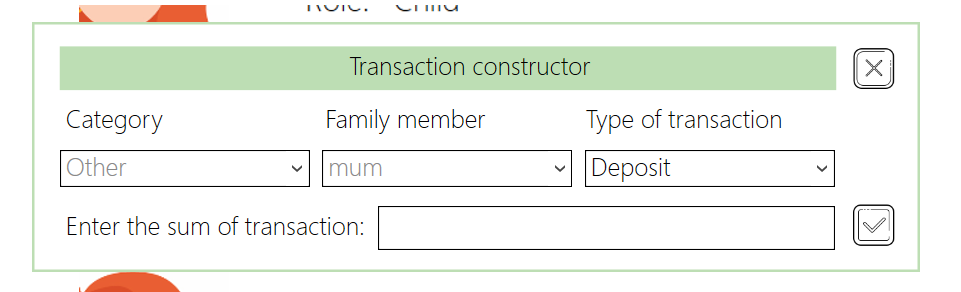


Рисунок 5.7 — Тип транзакции «Deposit»

Выбранная категория «Transfer» позволяет выбрать пользователя из списка.

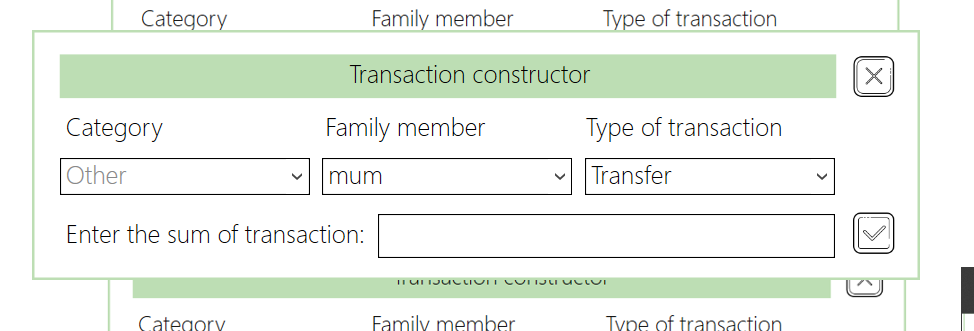


Рисунок 5.8 — Тип транзакции «Transfer»

В случае, если зарегистрирован только один пользователь, тип «Transfer» будет недоступен.

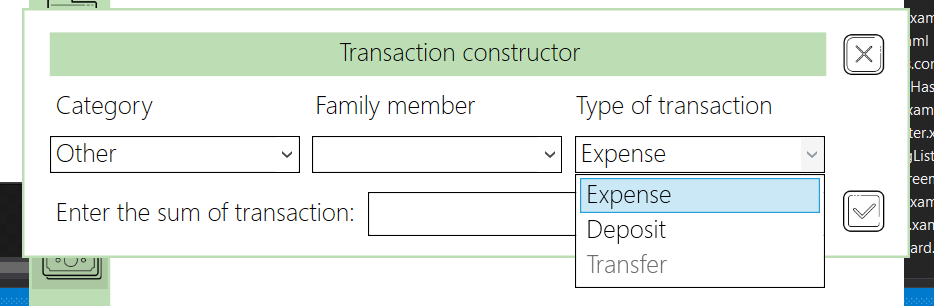


Рисунок 5.9 — Выбор типа транзакций, если зарегистрирован 1 пользователь

Если был авторизован пользователь с ролью «Ребенок», то тип транзакции «Deposit» заблокирован.

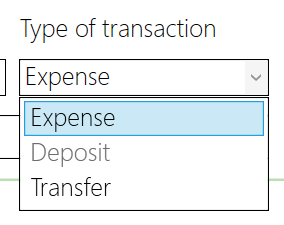
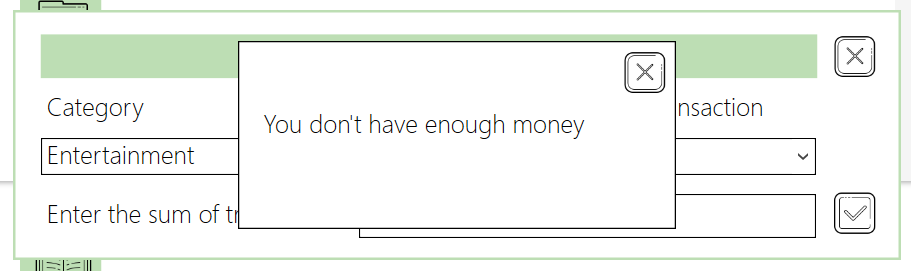


Рисунок 5.10 — Выбор типа транзакций, для пользователя, с ролью «Ребенок»

Если денег на счету у пользователя недостаточно для совершения транзакции, выводится сообщение об ошибке (рисунок 5.11).



Для обновления данных существует кнопка «Обновить». Для того, чтобы пользователь не забывал ее нажимать, после занесения изменений в базу данных выводится сообщение, приведенное на рисунке 5.12.

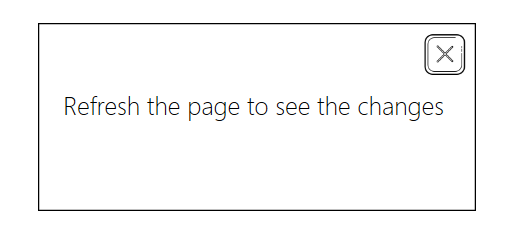


Рисунок 5.12 — Напоминание обновить страницу

На данном этапе были выполнены тесты на проверку работоспособности приложения, а именно на проверку валидации. Были проведены анализы результатов, которые показали, что валидация в приложении работает в соответствии с тем, как она задумывалась.

6 Руководство по использованию

После первого запуска приложения открывается окно с регистрационной формой для первого пользователя (рисунок 6.1), которому по умолчанию присваивается роль «Родитель». Это обусловлено тем, что у пользователя с ролью «Ребенок» нет возможности добавлять новых пользователей.

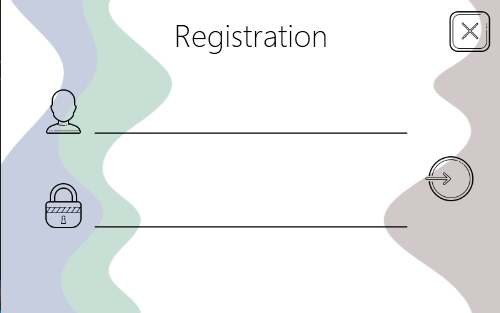


Рисунок 6.1 — Регистрационное окно

В дальнейшем, при открытии окна с регистрационной формой появляется возможность выбора роли пользователя (рисунок 6.2).

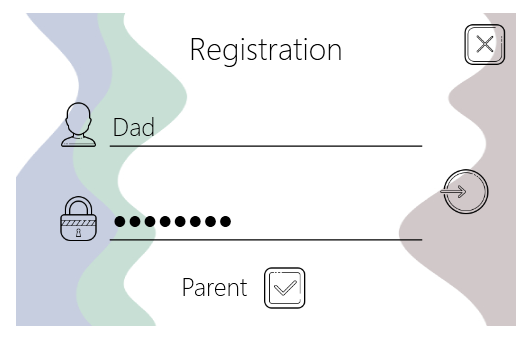


Рисунок 6.2 — Регистрационное окно

После успешного прохождения аутенфикации, пользователь будет перенаправлен на главную страницу (рисунок 6.3). В противном случае, на экране появится сообщение с соответствующим текстом.

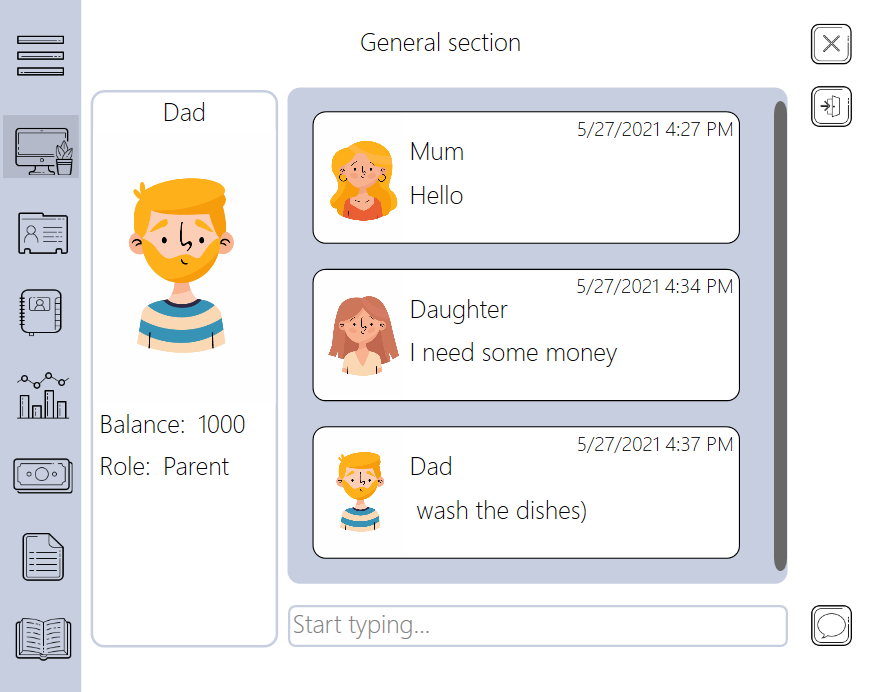


Рисунок 6.3 — Главная страница

На главной странице слева размещена краткая информация о пользователе: имя пользователя, аватар, баланс, роль.

Справа находятся комментарии. Самый новый комментарий располагается снизу. На комментарий накладывается ограничение по количеству символов — 20.

Переключение между вкладками осуществляется с помощью меню, расположенного в левой части окна.

Однако, не обязательно каждый раз открывать меню для перехода на новую вкладку. Достаточно просто нажать на соответствующую иконку. Меню пользователя с ролью «Ребенок» не содержит вкладок «Family» и «Statistics».

На рисунках 6.4 и 6.5 открыто меню с выбранной вкладкой «General» для родителя и ребенка соответственно.

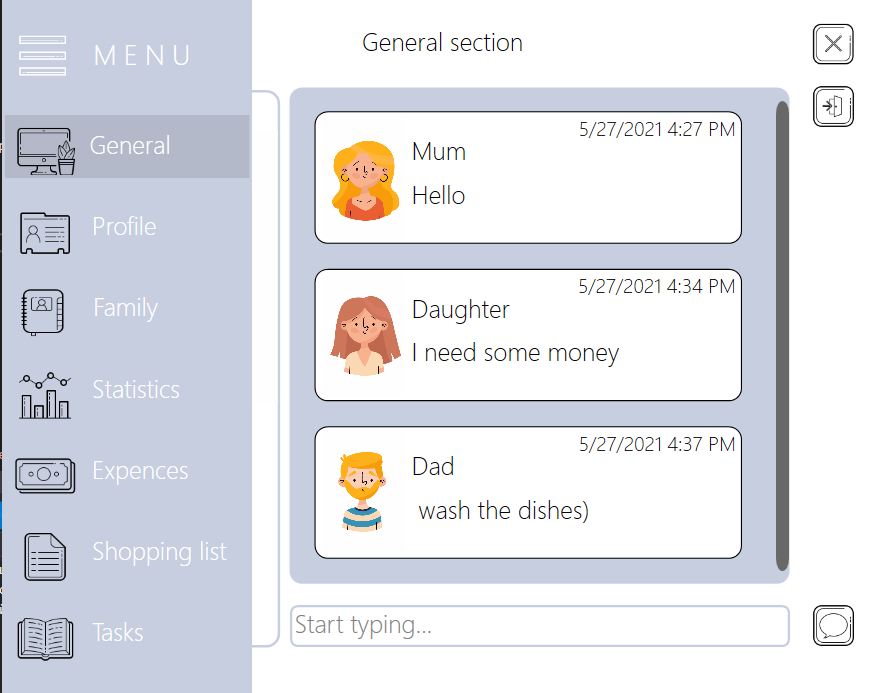


Рисунок 6.4 — Меню для пользователя с ролью «Родитель»

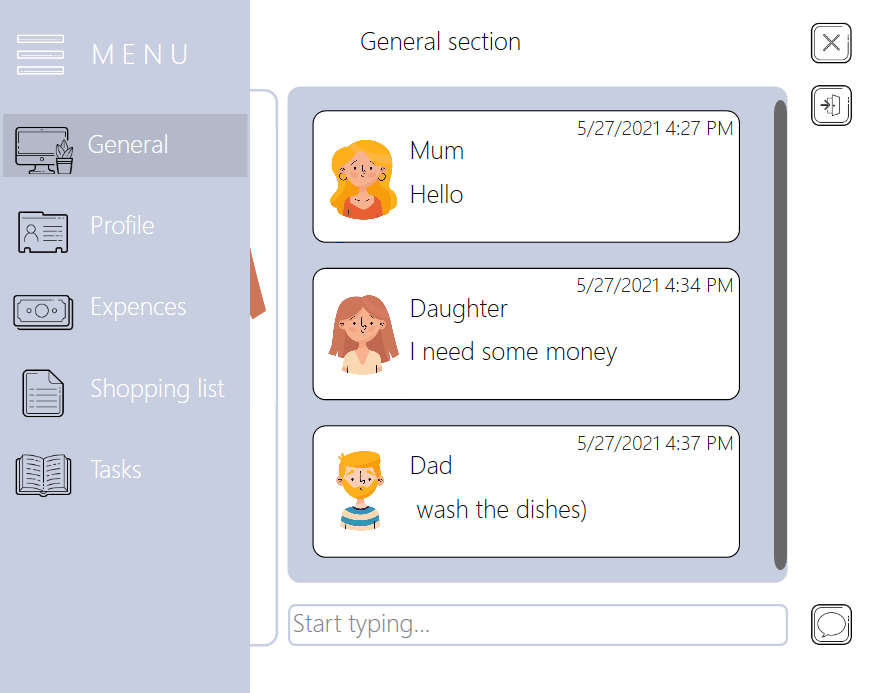


Рисунок 6.5 — Меню для пользователя с ролью «Ребенок»

На рисунке 6.5 представлена страница «Profile».

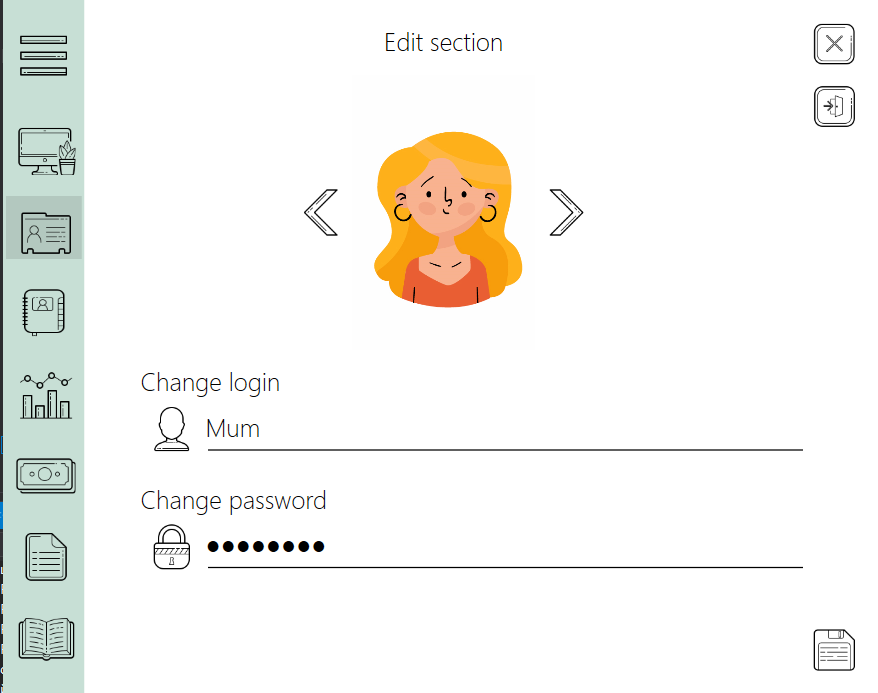


Рисунок 6.5 — «Profile»

Перейдя на страницу «Profile», пользователь получает возможность редактировать профиль: изменять фотографию, логин и пароль.

Изменения не сохраняются автоматически. Для того, чтобы сохранить изменения, необходимо нажать на кнопку «Сохранить» в правом нижнем углу.

Для того, чтобы изменить логин или пароль необходимо вписать новые значения в соответствующие поля.

Для просмотра и изменения возможных аватаров используются стрелки.

Страницы «Family» и «Statistics» доступны только администратору.

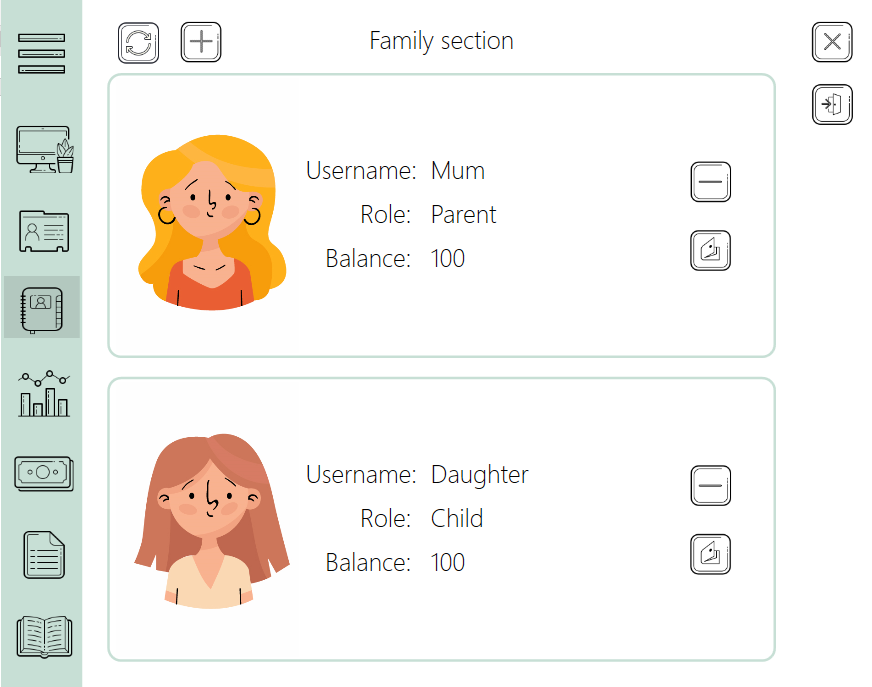


Рисунок 6.6 — «Family»

Страница «Family» содержит в себе список пользователей.

Используя кнопки, находящиеся напротив каждого пользователя, администратор имеет возможность удалять пользователей, а также пополнять их баланс.

С помощью кнопки «Добавить» в левом верхнем углу администратор может добавлять новых пользователей. При этом открывается окно с регистрационной формой, изображенное на рисунке 6.2.

Список пользователей не обновляется автоматически. Для этого предусмотрела кнопка «Обновить» в левом верхнем углу.

Страница «Expenses» предназначена для просмотра списка всех выполненных операций.

Она отличается для администратора и обычного пользователя тем, что обычный пользователь видит лишь свои операции, в то время как администратор может видеть операции всех пользователей, включая себя.

По центру, в верхней части страницы расположена форма выбора месяца, по которому выводится список операций. Доступные в форме месяцы заполняются из базы данных в зависимости от того, в каких месяцах была произведена по меньшей мере одна операция.

На рисунках 6.7 и 6.8 изображены страницы «Expenses» для родителя и ребенка соответственно.

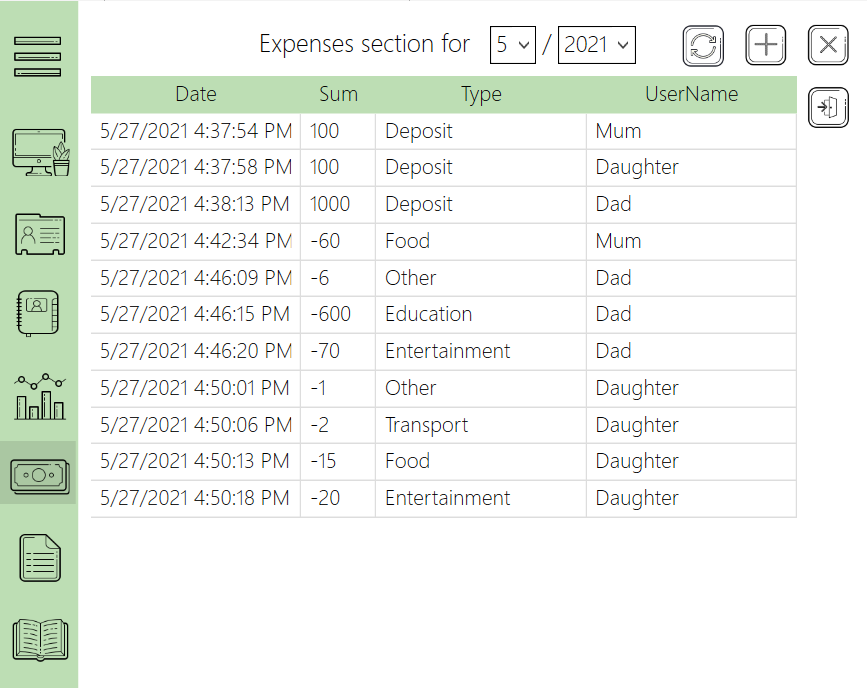


Рисунок 6.7 — «Expenses», роль – «Родитель»

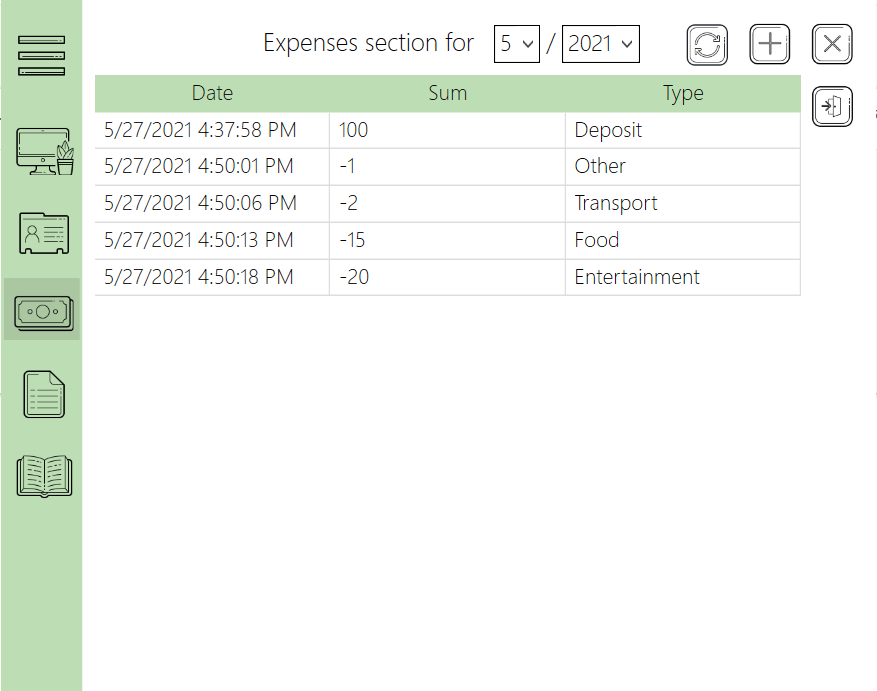


Рисунок 6.8 — «Expenses», роль – «Ребенок»

Справа, в верхней части находится кнопка «Добавить», при нажатии на которую на экран выводится окно с формой для добавления операции.

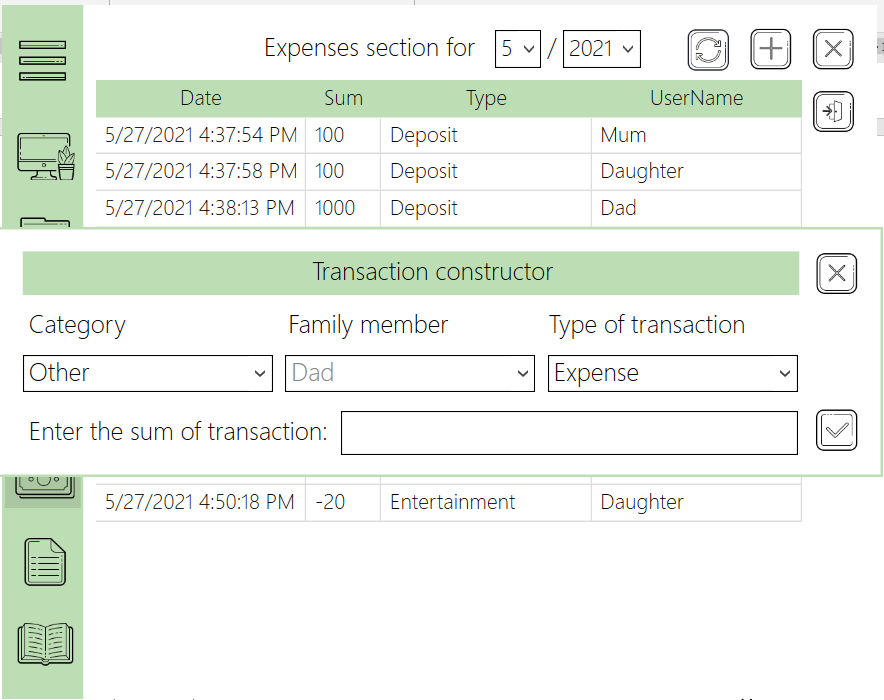


Рисунок 6.9 — «Expenses» добавление транзакции

На данной форме обычному пользователю недоступен тип операции «Deposit».

При выборе администратором типа операции «Deposit» остальные поля (кроме поля ввода суммы операции) становятся недоступными для ввода.

При выборе типа операции «Expense», доступным для ввода становится поле с категорией расходов.

При выборе типа операции «Transfer», доступным для ввода становится поле с выбором получающего пользователя.

Страница «Statistics» представлена на рисунке 6.10.

Страница доступна для просмотра только администраторам. Подобно странице «Expenses» в верхней части страницы располагается форма для выбора месяца, по которому выводится статистика.

На динамически обновляемом графике отображен баланс всех членов семьи в зависимости от времени.

На круговой диаграмме отображены основные статьи семейных расходов по категориям. При наведении на отдельный сектор рядом с курсором появляется небольшая подсказка, на которой отображён точный процент данной категории расходов от общих расходов.

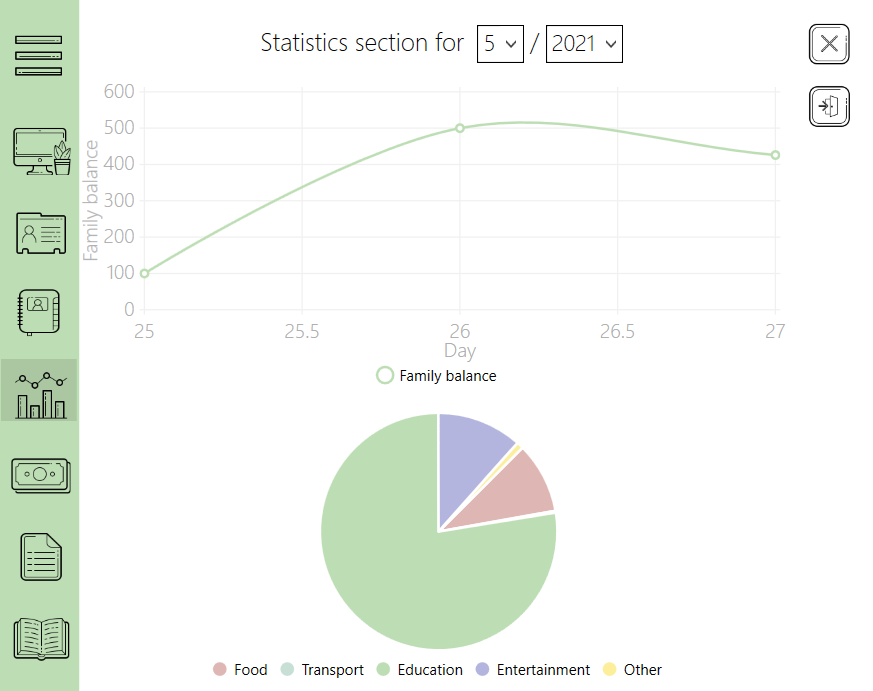


Рисунок 6.10 — «Expenses» добавление транзакции

Страница «Shopping list» имеет различный функционал для администратора и обычного ползователя.

Так, при добавлении элемента в список покупок с аккаунта администратора, он автоматически заносится в список покупок, в то время как при добавлении элемента в список покупок с аккаунта обычного пользователя, он попадает в список предложений.

Для того, чтобы этот элемент попал в настоящий список покупок, его должен принять администратор, используя кнопку «Принять» напротив каждого предложения в списке предложений, располагающемся в правой части страницы.

На рисунках 6.11 и 6.12 изображены страницы «Expenses» для родителя и ребенка соответственно.

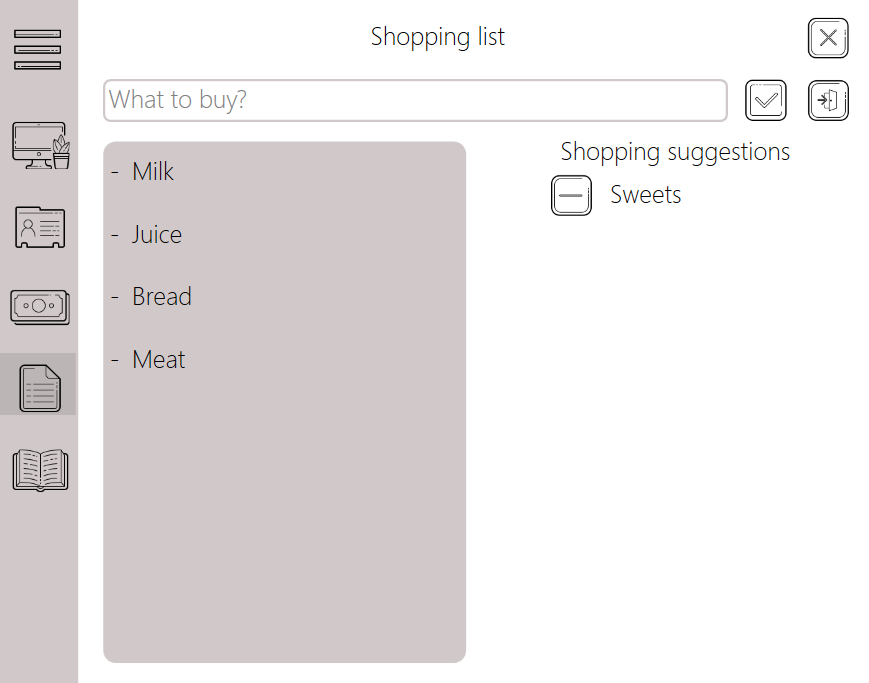


Рисунок 6.11 — «Shopping list», роль – «Ребенок»



Рисунок 6.12 — «Shopping list», роль – «Pодитель»

Страница «Diary», изображенная на рисунке 6.13, предназначена для составления планов на день.

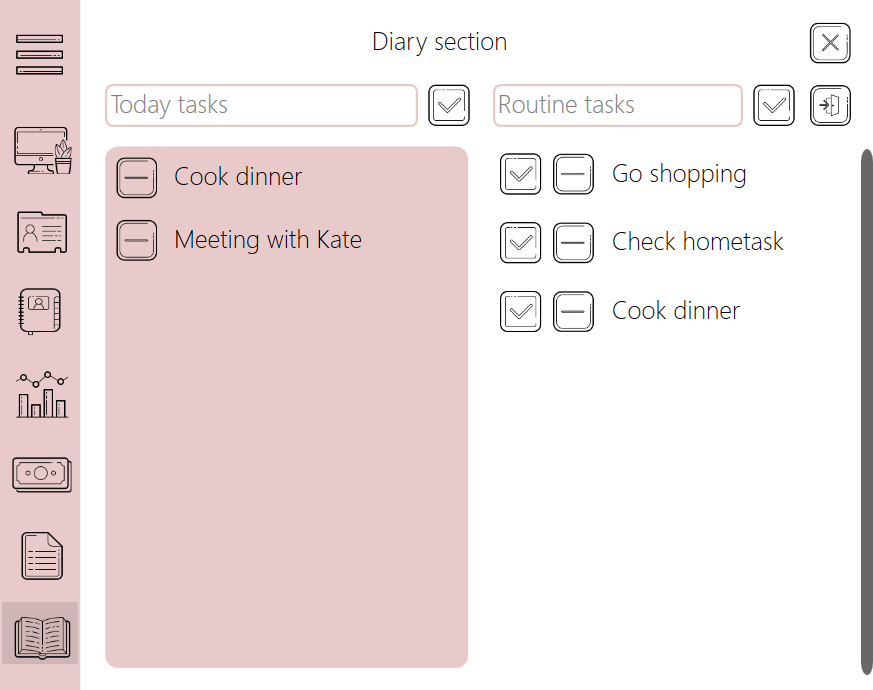


Рисунок 6.11 — «Diary»

На странице расположены 2 поля ввода: для непосредственного добавления заданий и для добавления частых заданий.

Задания, расположенные справа, можно добавить в спилок слева нажатием на кнопку «Добавить». После нажатия на кнопку «Удалить» в основном списке заданий в случае, если задание было из списка слева, оно не пропадает навсегда.

Чтобы удалить задание из списка справа навсегда, нужно нажать кнопку «Удалить» в списке слева.

# Заключение

В данном курсовом проекте было разработано программное средство «Family Organizer» при помощи языка программирования C#, API-интерфейса Windows Presentation Foundation, технологии Entity Framework и базы данных SQLite. При выполнении курсового проекта использовались принципы и приемы ООП.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* вся информация хранится в базе данных;
* приложение производит валидацию вводимых пользователем данных;
* приложение корректным образом обрабатывает возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение предоставляет пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение предоставляет возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* приложение предоставляет возможность просмотра статистики бюджета по заданному месяцу;
* приложение предоставляет возможность составления списка покупок, планов на день;
* приложение предоставляет возможность оставлять краткие комментарии на главной странице;
* приложение предоставляет возможность редактирования профиля;
* приложение предоставляет инструменты для контроля семейного бюджета.

Для хранения исходного кода и удобства контроля версий проекта использовался крупнейший веб-сервис для совместной разработки – GitHub. Ссылка на проект: (<https://github.com/AnastasiaPerkal/FamilyOrganizer>).

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

Список использованных источников

1. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2018. – 175 с.

2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.

3. Официальная документация Microsoft [Электронный ресурс] / Режим доступа: http://docs.microsoft.com/

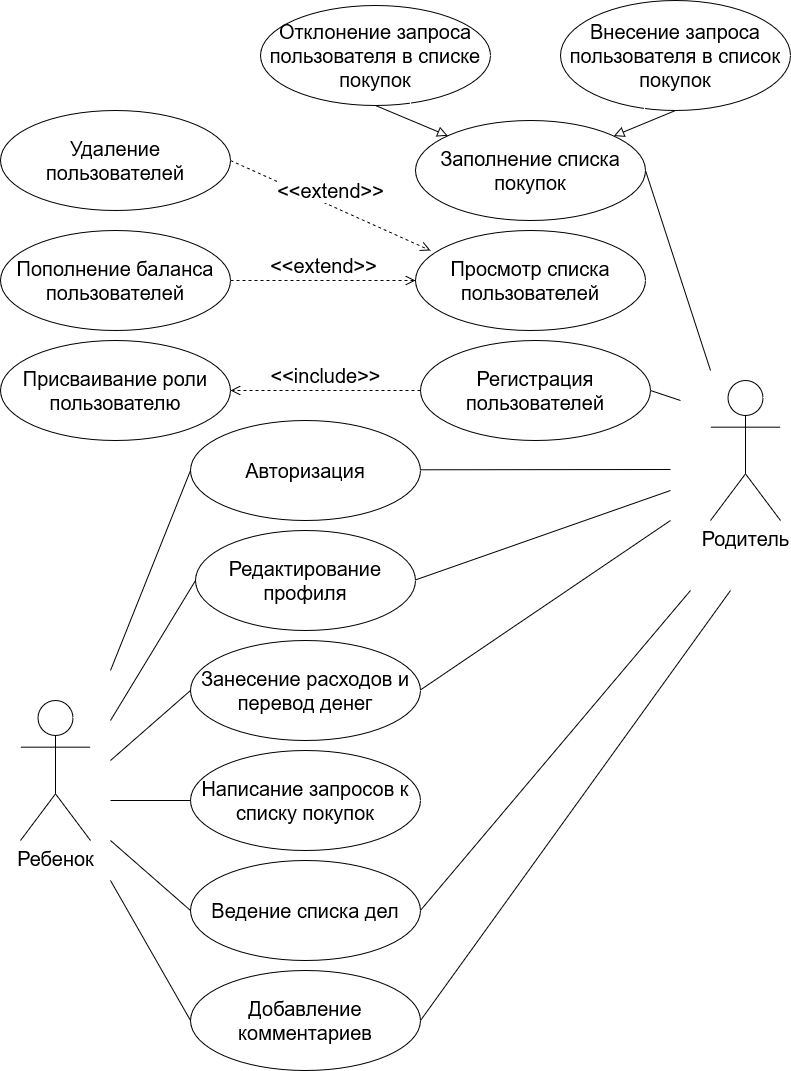
4. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru

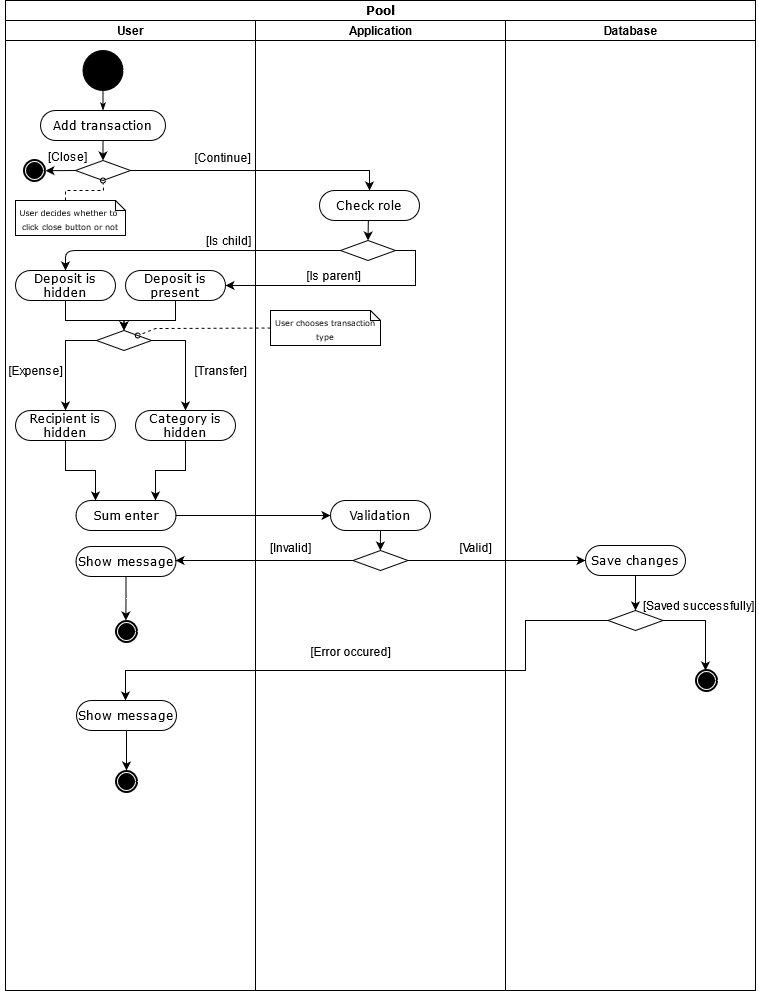
6. StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>

7. Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>

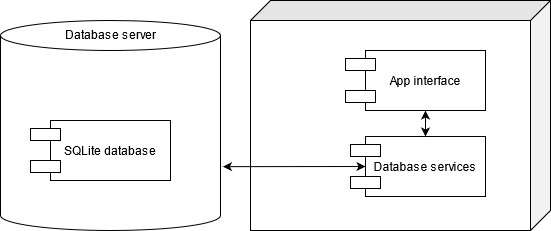
Приложение А



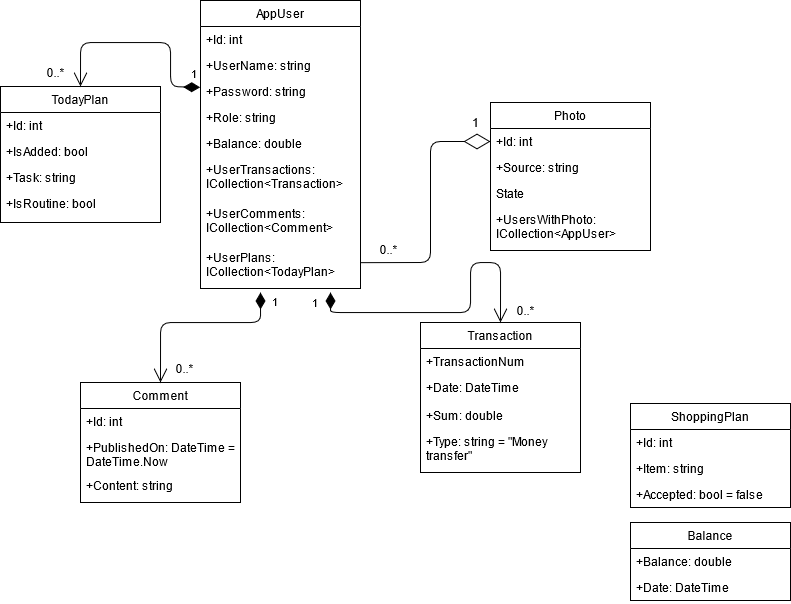
Приложение Б



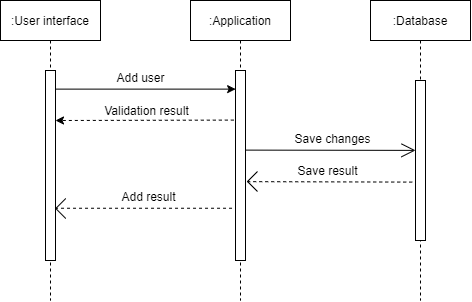
Приложение В



Приложение Г



Приложение Д



Приложение Е

Листинг 1 — Настройка класса контекста с помощью Fluent API

|  |
| --- |
|  |
|  |

Приложение Ж

Листинг 1 — Код, выполняющий хэширование паролей

|  |
| --- |
|  |

Листинг 2 — Код, выполняющий сравнение хэшей

|  |
| --- |
|  |

Приложение З

Листинг 1 — Переключение между страницами приложения

|  |
| --- |
|  |