МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных финансового учёта с применением технологии средств мониторинга состояния СУБД»

Выполнил студент Дмитрук Илья Игоревич

(Ф.И.О.)

Руководитель работы асс. Нистюк О.А.

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

И.о. зав. кафедрой ст. преп. Блинова Е.А.

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

Курсовая работа защищена с оценкой

Минск 2023

**Содержание**

[Введение 3](#_Toc153353371)

[1 Постановка задачи 4](#_Toc153353372)

[1.1 Аналитический обзор аналогов и литературных решений 4](#_Toc153353373)

[1.2 Анализ и разработка функциональных требований 6](#_Toc153353374)

[2 Проектирование базы данных 8](#_Toc153353375)

[3 Разработка необходимых объектов 11](#_Toc153353376)

[3.1 Таблицы 11](#_Toc153353377)

[3.2 Пользователи и роли 12](#_Toc153353378)

[3.3 Процедуры 14](#_Toc153353379)

[4 Описание процедур импорта и экспорта 16](#_Toc153353389)

[4.1 Процедура импорта данных в БД 16](#_Toc153353390)

[4.2 Процедура экспорта данных из БД 16](#_Toc153353391)

[5 Тестирование производительности 17](#_Toc153353392)

[6 Описание технологии применяемой в БД 19](#_Toc153353393)

[6.1 Словари 19](#_Toc153353394)

[6.2 Instance Viewer 19](#_Toc153353395)

[7 Краткое описание приложения 21](#_Toc153353396)

[8 Руководство пользователя 22](#_Toc153353397)

[Заключение 27](#_Toc153353398)

[Список используемых источников 29](#_Toc153353399)

[Приложение А 30](#_Toc153353400)

[Приложение Б 32](#_Toc153353401)



# Введение

Цель данной работы заключается в создании реляционной базы данных для приложения финансовый помощник, которое позволит пользователям удобно вести учёт за финансами и следить за ними.

База данных – это организованное собрание данных, которое обычно хранится в электронном виде в компьютерной системе. БД используются для хранения, организации и управления большим объемом структурированных и неструктурированных данных. Реляционная база данных является наиболее распространенной формой организации данных, в которой данные представлены в виде таблиц, состоящих из строк и столбцов, где каждый столбец представляет атрибут, а каждая строка представляет кортеж или запись. В данной работе для управления базой данных была выбрана СУБД Oracle, поскольку эта система обладает большим функционалом, что позволяет обеспечить эффективное хранение, обработку и управление данными.

Также необходимо разработать приложение для демонстрации функциональности базы данных и взаимодействия с ней. Приложение было реализовано с использованием Windows Presentation Foundation (WPF) на языке C#.

Для анализа состояния базы данных, в курсовой работе применяются различные способы мониторинга состояния СУБД.

Основные требования к приложению:

* реализация ролей администратора, пользователя;
* добавление, удаление сохранения(дохода/расхода);
* добавление, удаление, изменение категорий;
* подсчёт суммы доходов/расходов за период;
* подсчёт частоты доходов/расходов по категориям;
* подсчёт общей суммы бюджета;
* подсчёт общей суммы бюджета;
* подсчёт суммы долга;
* подсчёт суммы выплаченного долга;
* обеспечение резервирования сохранений;
* взаимодействие с БД при помощи хранимых процедур.

Содержание данной пояснительной записки отражает этапы выполнения курсового проекта.

1. Постановка задачи

Задачи проекта: разработать архитектуру базы данных, создать процедуры, взаимодействие с которыми будет понятно любому пользователю.

Цели приложения состоит в демонстрации возможности базы данных и предоставления удобного инструмента для ведения финансового учёта

В соответствии с заданием курсового проекта следует не только создать базу данных, но и разработать программное средство, которое должно в полной мере демонстрировать возможности базы данных.

Для того, чтобы сформировать окончательные требования к проектируемому программному средству сначала рассмотрим прототипы из той же области.

* 1. Аналитический обзор аналогов и литературных решений

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

На сегодняшний день аналогичных приложений или web-сервисов для контроля базы данных финансового учёта в открытом доступе не много. Поэтому было рассмотрено всего пара аналогов.

Money Love – это мобильное приложение, предназначенное для слежения за своими расходами. Приложение всю информацию о расходах отображает визуально. Можно просмотреть информацию в виде диаграмм, круговых диаграмм, графика. Диаграммы отображают то, какую часть расходов пользователь потратил на определённую категорию. График показывает рост или снижение расходов. При сохранении расхода, можно указать много различной информации. Однако в приложении много элементов интерфейсов, которые занимают много места, не соразмерного их смысловой нагрузке. Интерфейс приложения представлен на рисунке 1.1

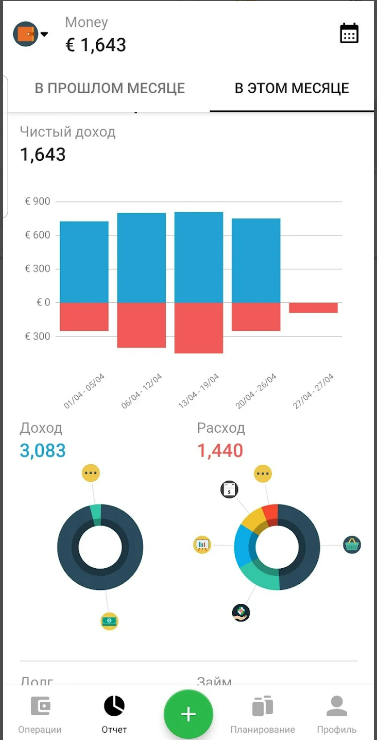


Рисунок 1.1 – Интерфейс приложения «Money Love»

Проанализировав «Money Love», можно выделить основные минусы и плюсы данного программного средства.

Основные плюсы:

* сохранённая информация отображается визуально;
* можно сохранять много различной информации.

Основные минусы:

* простой дизайн;
* много ненужных элементов, занимающих много место.

Веб–сайт Finmap представляет из себя интернет-ресурс для слежения за своими доходами и расходами. На главной странице присуствует список с историей доходов/расходов. В списке мы можем изменять уже сохранённую информацию. Таким образом, если мы сохраним что-то не так, это всегда можно исправить. Дизайн веб-сайта сделан неплохо, хорошо сочетаются цвета, так, например, всё что связанно с доходами, отображенно оттенками зелёного цвета, а то что связанно с расходами, красного. Однако, в плане визуального отображения информации, присуствуют явные недостатки. Так, например, на графиках, отображенно очень много различной информации, из-за чего, общая информация плохо воспренимается. Интерфейс веб-сайта на рисунке 1.2.

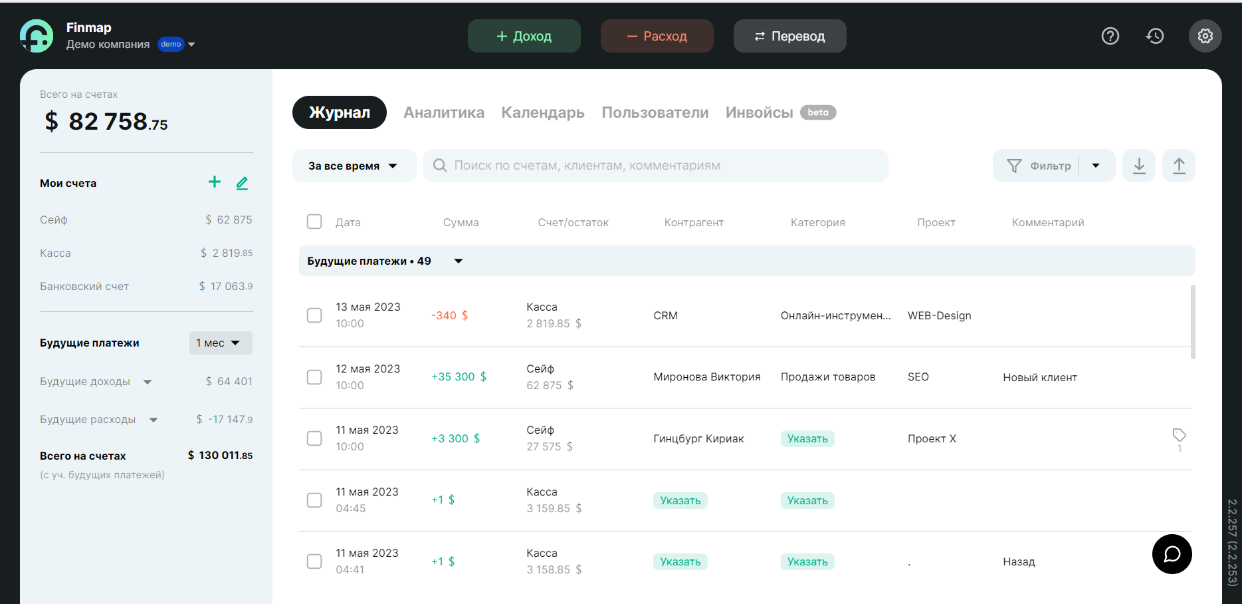


Рисунок 1.2 ­– интерфейс веб-сайта «Finmap»

Проанализировав «Finmap», можно выделить основные минусы и плюсы данного программного средства.

Основные плюсы:

* хорошее сочетание цветов;
* удобный интерфейс;
* можно редактировать сохранённую информацию.

Основные минусы:

* отображается слишком много ненужной информации.

Таким образом был выполнен анализ необходимого функционала, предоставленный аналогами разрабатываемого приложения.

* 1. Анализ и разработка функциональных требований

Анализ требований – процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки ПО.

Цель анализа требований в проектах – получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные сопутствующие риски.

На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи создаваемого проекта.

Определение и описание требований – шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга.

Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований будут рассматривать общие требования к разрабатываемому курсовому проекту.

К их этим требованиям относятся:

* использование системы управления базами данных (СУБД);
* доступ к данным должен осуществляться только через соответствующие хранимые процедуры.

Следующими требованиями являются требования пользователей приложения, а именно администратора и пользователя.

Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы.

Пользователь данного программного решения должен иметь возможности, соответствующие его роли.

Возможности администратора:

* управление базой данных категорий;
* управление базой данных пользователей.

Возможности пользователя:

* добавления сохранения (дохода/расхода);
* удаление сохранения (дохода/расхода);
* добавления цели;
* удаление цели;
* редактирование цели;
* сохранение информации о вложении в цель;
* просмотр прогресса достижения целей;
* добавления долга;
* удаление долга;
* редактирование долга;
* сохранение информации о оплате долга;
* просмотр прогресса выплаты долгов.

После проведения анализа были выявлены следующие функциональные требования программного средства:

* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем различных данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* Должен производиться анализ финансового состояния пользователя.

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.

1. Проектирование базы данных

Проектирование баз данных представляет собой процесс создания структуры базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных из таблиц по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проходит через два основных этапа: концептуальное и логическое проектирование.

Концептуальное проектирование представляет собой создание семантической модели предметной области, то есть информационной модели на высоком уровне абстракции.

Концептуально на данном этапе было выделено 5 сущностей:

* пользователь;
* сохранение (доход/расход);
* категория;
* цель;
* долг.

Так же были определены необходимые связи. Например, между сущностями «пользователь» и «сохранение» установлена связь один-ко-многим. Так же были выделены атрибуты для сущностей. Например, для сущности «пользователь» были выделены атрибуты такие, как «логин», «имя», «изображение», «пароль», «роль».

Для взаимодействия с таблицами и приложением были определены специальные пользователи «администратор», «пользователь».

Логическое проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных.

Для реляционной модели данных 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.

Логическая модель базы со структурой связей представлена на рисунке 2.1. В ней показано как будут взаимодействовать таблицы между собой.

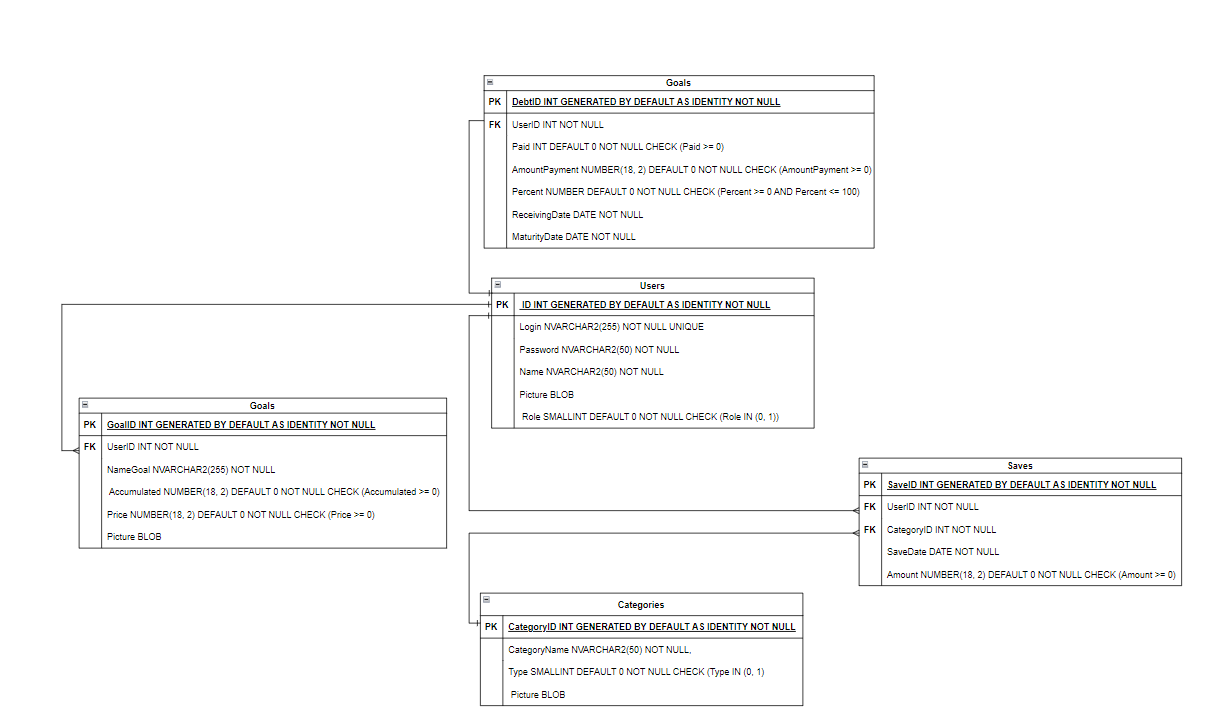


Рисунок 2.1 ­– Логическая модель базы данных

Всего в базе данных содержится 5 таблиц.

Таблица Users предназначена для хранения данных зарегистрированных пользователей в базе данный.

Таблица Saves предназначена для хранения сохранений(доходов/расходов).

Таблица Categories предназначена для хранения данных категорий доходов/расходов.

Таблица Goals предназначена для хранения данных целей.

Таблица Debts предназначена для хранения данных долгов.

Так же была разработана use-case диаграмма, предоставленная на рисунке 2.2.



Рисунок 2.2 – Use-case диаграмма

Всего в базе данных 2 пользователя:

Пользователь ADMIN является служебным пользователем необходимым для авторизации и регистрации.

Пользователь ADMIN создан для представления администратора в базе данных. Он предназначен для редактирования категорий и пользователей, а также обеспечивает возможность авторизации.

Пользователь FA\_USER создан для пользователя. Данный пользователь имеет возможность добавлять и удалять сохранения, управлять своими целями, просматривать прогресс их достижения, управлять своими долгами и просматривать прогресс выплаты их, редактировать свои данные.

1. Разработка необходимых объектов
   1. Таблицы

Таблица – это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Она состоит из столбцов и строк.

Столбцы таблицы называют полями; каждое поле характеризуется своим именем (названием соответствующего свойства) и типом данных, отражающих значения данного свойства. Каждое поле обладает определенным набором свойств (размер, формат и др.).

Поле базы данных – это столбец таблицы, включающий в себя значения определенного свойства.

В каждой таблице должно быть, по крайней мере, одно ключевое поле, содержимое которого уникально для любой записи в этой таблице.

Значения ключевого поля однозначно определяют каждую запись в таблице.

Для реализации базы данных «Реализация базы данных финансового учёта с применением технологии средств мониторинга состояния СУБД» было разработано 5 таблиц: Users, Saves, Categories, Goals, Debts. Они будут реализованы в СУБД Oracle DataBase 23c.

Таблица Users представляет список всех зарегистрированных пользователей в базе данных (Таблица 3.1).

Таблица 3.1 – Столбцы таблицы Users

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор пользователя | int |
| Login | Логин пользователя | nvarchar2 |
| Name | Имя пользователя | nvarchar2 |
| Password | Пароль пользователя | nvarchar2 |
| Picture | Изображение пользователя | blob |
| Role | Телефонный номер пользователя | smallint |

Таблица Saves представляет список всех сохранений (таблица 3.2).

Таблица 3.2 – Столбцы таблицы Saves

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| SaveID | Идентификатор сохранения | int |
| Amount | Сумма сохранения | number |
| SaveDate | Дата сохранения | date |
| CategoryID | Идентификатор категории | int |
| UserID | Идентификатор пользователя | int |

Таблица Categories представляет список всех категорий (таблица 3.3).

Таблица 3.3 – Столбцы таблицы Categories

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| CategoryID | Идентификатор категории | int |
| CategoryName | Название категории | nvarchar2 |
| Type | Тип категории (доход/расход) | smalllint |
| Picture | Изображение категории | blob |

Таблица Goals представляет список всех целей (таблица 3.4).

Таблица 3.4 – Столбцы таблицы Goals

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| GoalID | Идентификатор цели | int |
| NameGoal | Название цели | nvarchar2 |
| Accumulated | Сколько уже накоплено на цель | number |
| Price | Цена цели | number |
| Picture | Изображение цели | blob |
| UserID | Идентификатор пользователя | int |

Таблица Debts представляет список всех долгов (таблица 3.5).

Таблица 3.5 – Столбцы таблицы Debts

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| DebtID | Идентификатор долга | int |
| NameDebt | Название долга | nvarchar2 |
| Paid | Количество совершённых платежей | int |
| AmountPayment | Сумма платежа | number |
| NumberPayments | Количество платежей | int |
| Percent | Процент долга | number |
| ReceivingDate | Дата получения долга | date |
| MaturityDate | Предполагаемая дата выплаты долга | date |
| UserID | Идентификатор пользователя | int |

Скрипты создания таблиц и наложение ограничений целостности на них представлены в приложении А данной записки.

* 1. Пользователи и роли

База данных содержит 2 основных пользователя: ADMIN и FA\_USER. Пользователь ADMIN представляет собой хранилище всех объектов базы данных. Он занимается созданием инфраструктуры, объектов БД и управлением приложения. USER может взаимодействовать с базой данных благодаря полномочиям на вызов процедур.

Для каждого пользователя была создана роль. Роль - поименованный набор привилегий. В пример создания роли предоставлен листинг 3.2 где создаётся роль для пользователя USER.

CREATE ROLE UserRole;

GRANT CREATE SESSION TO UserRole;

GRANT EXECUTE ON ADMIN.GetUserByID TO UserRole;

GRANT EXECUTE ON ADMIN.AlterUser TO UserRole;

GRANT EXECUTE ON ADMIN.AlterUserByNameAndPicture TO UserRole;

GRANT EXECUTE ON ADMIN.GetAllSaves TO UserRole;

GRANT EXECUTE ON ADMIN.GetSavesByCategoryType TO UserRole;

GRANT EXECUTE ON ADMIN.AddSaves TO UserRole;

GRANT EXECUTE ON ADMIN.DeleteSave TO UserRole;

GRANT EXECUTE ON ADMIN.GetCategoriesByType TO UserRole;

GRANT EXECUTE ON ADMIN.GetAllGoals TO UserRole;

GRANT EXECUTE ON ADMIN.AddGoal TO UserRole;

GRANT EXECUTE ON ADMIN.AlterGoal TO UserRole;

GRANT EXECUTE ON ADMIN.DeleteGoal TO UserRole;

GRANT EXECUTE ON ADMIN.GetAllDebts TO UserRole;

GRANT EXECUTE ON ADMIN.GetDebtByID TO UserRole;

GRANT EXECUTE ON ADMIN.AddDebt TO UserRole;

GRANT EXECUTE ON ADMIN.AlterDebt TO UserRole;

GRANT EXECUTE ON ADMIN.DeleteDebt TO UserRole;

GRANT EXECUTE ON ADMIN.CalculateSaves TO UserRole;

GRANT EXECUTE ON ADMIN.CalculateBudget TO UserRole;

GRANT EXECUTE ON ADMIN.CalculateSavesByCategory TO UserRole;

GRANT EXECUTE ON ADMIN.CalculateTotalDebt TO UserRole;

GRANT EXECUTE ON ADMIN.CalculateTotalDebts TO UserRole;

GRANT EXECUTE ON ADMIN.CalculatePaidDebt TO UserRole;

GRANT EXECUTE ON ADMIN.CalculatePaidDebts TO UserRole;

Листинг 3.2 – Создание роли для пользователя USER

После создания роли и назначения ей привилегий, необходимо создать пользователя и присвоить ему роль, созданную ранее. Создание пользователя FA\_USER приведено в листинге 3.3.

CREATE USER FA\_USER IDENTIFIED BY 1234;

GRANT UserRole TO FA\_USER;

ALTER USER FA\_USER QUOTA UNLIMITED ON FA\_TS;

Листинг 3.3 – Создания пользователя FA\_USER

Пользователю ADMIN создаётся по создаётся по тому же принципу, но ему присваивается привилегия DBA.

* 1. Процедуры

Для управления данными, пользователи вызывают процедуры. Процедура – это предварительно скомпилированный блок инструкций SQL, который сохранен в базе данных для последующего выполнения. В нашей базе данных процедуры создаются у пользователя ADMIN, а пользователю FA\_USER, предоставляется к ним доступ.

Процедура GetAllUsers предназначена для получения всех пользователей.

Процедура GetUserByID предназначена для получения пользователя по идентификатору.

Процедура GetUserByLoginAndPassword предназначена для проведения авторизации и получения пользователя по логину и паролю.

Процедура AddUser предназначена для добавления пользователя.

Процедура AlterUser предназначена для изменения данных пользователя.

Процедура AlterUserWithoutPassword предназначена для изменения данных пользователя, не затрагивая его пароль.

Процедура AlterUserByNameAndPicture предназначена для имени и изображения пользователя.

Процедура DeleteUser предназначена для удаления пользователя.

Процедура GetAllSaves предназначена для получения всех сохранений.

Процедура GetSavesByCategoryType предназначена для получения всех сохранений с категорией определённого типа (дохода/расхода).

Процедура AddSaves предназначена для добавления сохранения.

Процедура DeleteSave предназначена для удаления сохранения.

Процедура GetCategoriesByType предназначена для получения всех с категорий определённого типа (дохода/расхода).

Процедура AddCategory предназначена для добавления категории.

Процедура AlterCategory предназначена для изменения данных категории.

Процедура DeleteCategory предназначена для удаления категории.

Процедура GetAllGoals предназначена для получения всех целей.

Процедура AddGoal предназначена для добавления цели.

Процедура AlterGoal предназначена для изменения данных цели.

Процедура DeleteGoal предназначена для удаления цели.

Процедура GetAllDebts предназначена для получения всех долгов.

Процедура GetDebtByID предназначена для получения долга по идентификатору.

Процедура AddDebt предназначена для добавления долга.

Процедура AlterDebt предназначена для изменения данных долга.

Процедура DeleteDebt предназначена для удаления долга.

Процедура CalculateSaves предназначена для подсчёта суммы доходов/расходов за определённый промежуток времени.

Процедура CalculateBudget предназначена для подсчёта суммы бюджета (разность между доходами и расходами) за определённый промежуток времени.

Процедура CalculateSavesByCategory предназначена для подсчёта суммы доходов/расходов по определённой категории, а также процента дохода/расхода в категории, от общей суммы доходов/расходов.

Процедура CalculateTotalDebt предназначена для подсчёта итоговой суммы долга (сумма всех платежей с процентами).

Процедура CalculateTotalDebtsпредназначена для подсчёта итоговой суммы всего долга.

Процедура CalculatePaidDebt предназначена для подсчёта суммы выплаченного долга.

Процедура CalculatePaidDebts предназначена для подсчёта всей суммы выплаченных долгов.

Табличное описание процедур приведено в приложении Б.

1. Описание процедур импорта и экспорта
   1. Процедура импорта данных в БД

Чтобы импортировать данный в БД был выбран формат JSON.

Для импорта данных создается процедура с названием LoadDataFromJsonSaves, которая принимает в параметрах значение с типом clob и вставляет полученные данные в таблицу Saves. Код создания процедуры json\_ LoadDataFromJsonSaves представлен в листинге 4.1.

CREATE OR REPLACE PROCEDURE LoadDataFromJsonSaves(p\_JsonData CLOB) AS

BEGIN

INSERT INTO ADMIN.Saves (Amount, SaveDate, CategoryID, UserID)

SELECT Amount, TO\_DATE(SaveDate, 'YYYY-MM-DD'), CategoryID, UserID

FROM JSON\_TABLE(

p\_JsonData,

'$[\*]'

COLUMNS (

Amount PATH '$.Amount',

SaveDate PATH '$.SaveDate',

CategoryID PATH '$.CategoryID',

UserID PATH '$.UserID'

)

);

COMMIT; -- Фиксация изменений

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLCODE || ' - ' || SQLERRM);

ROLLBACK;

RAISE;

END LoadDataFromJsonSaves;

Листинг 4.1 – Создание процедуры LoadDataFromJsonSaves

Вызов процедуры и передача данных происходит из приложения.

* 1. Процедура экспорта данных из БД

Для экспорта данных из БД был выбран формат JSON. Для того что бы экспортировать данные из БД в JSON были использованы стандартные инструменты среды разработки SQL Developer.

1. Тестирование производительности

Для тестирования производительности БД в таблицу SAVES было добавлено 100000 строк при помощи импорта, описанного ранее. Для того что бы провести тесты цены запроса и времени его выполнения были разработаны три select запроса затрагивающие выборку с ключевыми полями и без них. Тестирование производиться при помощи инструментов среды разработки SQL Developer.

Первый запрос, подремонтированный в листинге 5.1, необходим для теста стоимости и времени запроса при поиске сохранения с указанным идентификатором.

SELECT \* FROM SAVES WHERE saveID = 4789;

Листинг 5.1 – Первый тестовый запрос

Результат первого тестового запроса показал, что время запроса составляет около 7ms, а цена 1 что является хорошим результатом для поиска с ключевым или же индексированным полем в БД с 100000 строк. Результат выполнения предоставлен на рисунке 5.1.

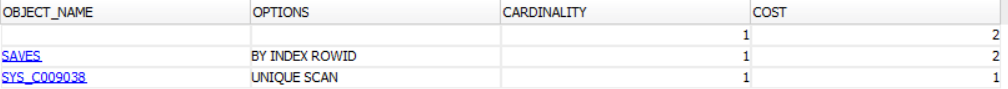


Рисунок 5.1 – Результат выполнения первого тестового запроса

Второй запрос, подремонтированный в листинге 5.2, необходим для теста стоимости и времени запроса при поиске строк c датами в определённом промежутке. Данный тест является важным ведь в приложении часто используется подобный поиск.

SELECT \* FROM SAVES WHERE SaveDate >= TO\_DATE('2023-12-01', 'YYYY-MM-DD') AND SaveDate <= TO\_DATE('2023-12-31', 'YYYY-MM-DD');

Листинг 5.2 – Второй тестовый запрос

Результат второго тестового запроса показал, что время запроса составляет около 56ms, а цена 2 что является хорошим результатом для поиска не ключевого или же индексированного поля в БД с 100000 строк. Результат выполнения предоставлен на рисунке 5.2.

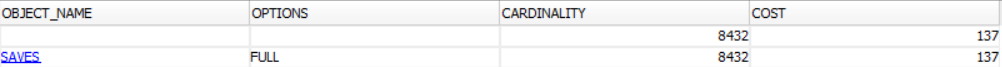


Рисунок 5.2 – Результат выполнения второго тестового запроса

Третий запрос, подремонтированный в листинге 5.3, необходим для теста стоимости и времени запроса при поиске строк с указанным идентификатором категории.

SELECT \* FROM SAVES WHERE CategoryID = 5;

Листинг 5.3 – Третий тестовый запрос

Результат третьего тестового запроса показал, что время запроса составляет около 6ms, а цена 137 что является хорошим результатом для поиска без ключевых или же индексированных полей в БД с 100000 строк. Результат выполнения предоставлен на рисунке 5.3.

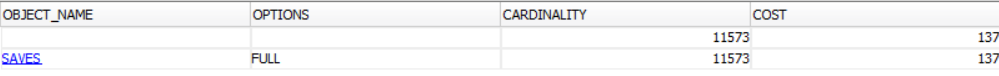


Рисунок 5.3 – Результат выполнения третьего тестового запроса

Как мы видим, несмотря на огромное количество строк, запросы выполняются довольно быстро, причём, мы даже не используем индексы в таблицы Saves.

1. Описание технологии применяемой в БД

В качестве основных технологий в БД были выбраны технологии средств мониторинга состояния СУБД. Мониторинг состояния СУБД – это процесс отслеживания и анализа различных аспектов работы базы данных с целью обеспечения её производительности, стабильности, доступности и эффективности. Мониторинг является важной частью администрирования баз данных и предоставляет информацию для выявления проблем, планирования масштабирования, оптимизации запросов и общего контроля за состоянием базы данных.

* 1. Словари

Для получения информации о состояния базы данных, можно использовать select запросы к специальным словарям.

Словарь V$INSTANCE хранит информацию об экземпляре базы данных, такую как имя экземпляра, версия Oracle, режим и другие параметры.

Словарь V$SQLSTATS хранит статистику по выполненным SQL-запросам. Он включает информацию о количестве выполненных запросов, времени выполнения, использовании ресурсов и других метриках.

Словарь V$SESSION хранит информацию о сеансах и активных подключениях к базе данных. Вы можете узнать о пользователях, статусе сеанса, времени выполнения и многом другом.

Словарь V$OSSTAT хранит информацию о системных метриках, таких как загрузка CPU, использование памяти и другие характеристики операционной системы.

Словарь V$SESSION\_EVENT хранит информацию о событиях, связанных с сеансами, такие как блокировки, ввод-вывод и других.

Словарь V$SYSTEM\_EVENT хранит информацию о событиях системы, которые могут влиять на производительность базы данных.

Словарь V$EVENT\_NAME хранит список доступных событий базы данных.

Словарь V$SYSTEM\_EVENT хранит статистику по использованию места в SGA.

Словарь V$FILESTAT хранит статистику использования файлов базы данных.

Словарь V$TEMPSTAT хранит статистику использования временных файлов.

* 1. Instance Viewer

Instance Viewer – это встроенный в SQL Developer инструмент, предназначенный для мониторинга состояния экземпляра базы данных Oracle. Его интерфейс представлен на рисунке 6.1.

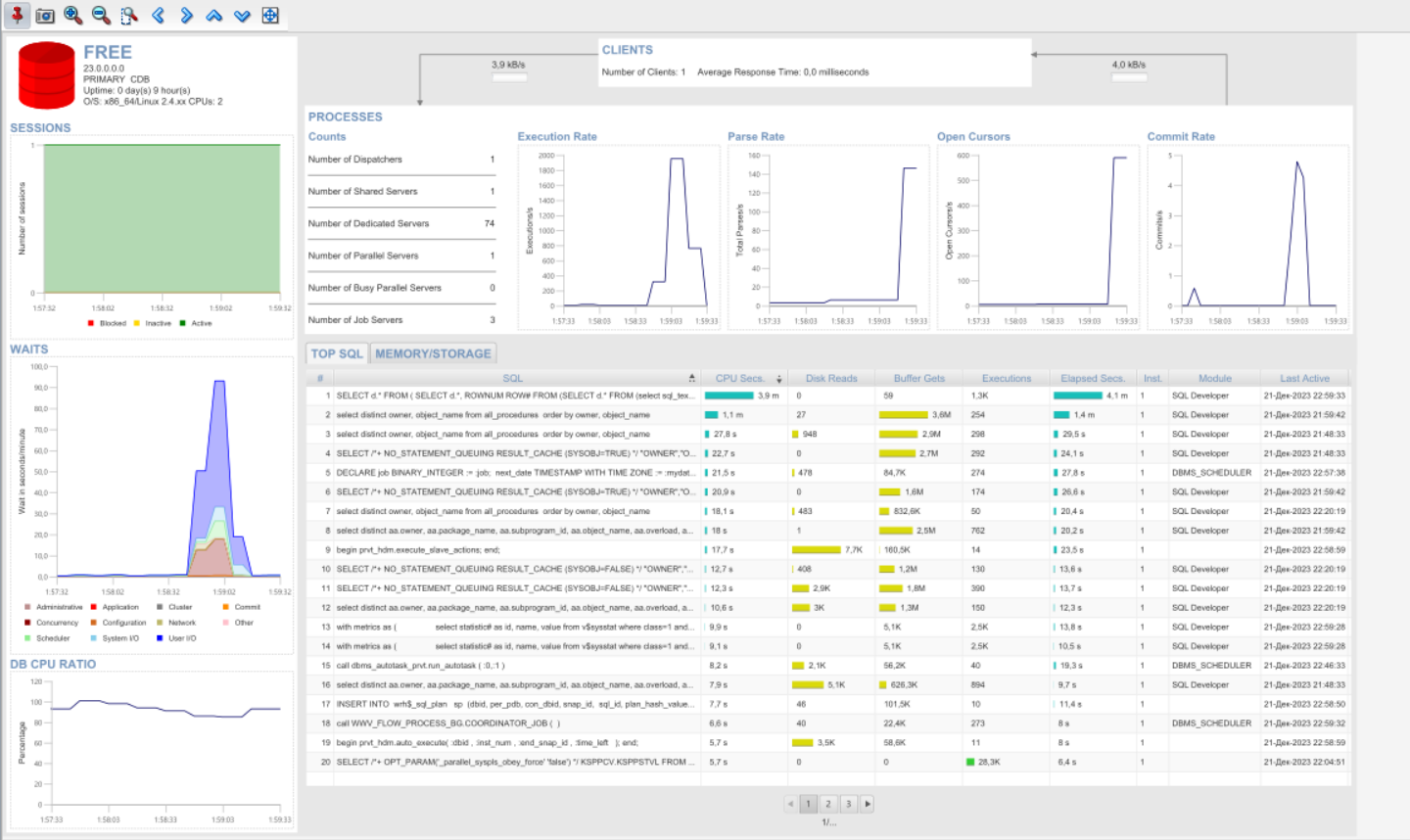


Рисунок 6.1 – Интерфейс Instance Viewer

Этот инструмент предоставляет графическое представление различных параметров, связанных с работой базы данных, таких как информация о сеансах, загруженности CPU, использования памяти, информация о запросах. Instance Viewer отслеживает активность и производительность базы данных в реальном времени.

1. Краткое описание приложения

Для демонстрации работы БД было разработано приложение, написанное на языке C# при помощи фреймворка WPF с использование языка разметки XAML. Для работы с БД были разработаны специальные классы: DBControler, AdditionsDB, CatigoriesDB, DebtDB, GoalsDB, UsersDB.

Класс DBControler является основным для работы с базой данных. В нём хранится объекты других классов, предназначенных для работы с БД. Эти классы содержат методы для, вызывающие определённые процедуры, взаимодействующие с определёнными таблицами. Так же в этом классе выполняется подключение к базе данных.

Для сопоставления данных из БД в приложение были созданные специальные классы модели: Addition, Categories, Debt, Goals, User.

1. Руководство пользователя

Разработанное в рамках курсового проекта приложение поддерживает две основные роли: администратор и пользователь. Заходя в приложение, пользователю открывается окно авторизации. Здесь он может выбрать вход или регистрацию. Если пользователь выбрал вход, то ему продолжится ввести логин и пароль. Если он всё введёт верно, то его перекинет в основное окно приложения. Если пользователь выберет регистрацию, то ему продолжится вести логин, имя и два раза пароль. После подтверждения регистрации, в случае если пользователь с введённым логином не регистрировался раньше, его введённые данные сохраняться в БД, а его самого перекинет в основное окно приложения. Скриншот окна авторизации представлен на рисунке 8.1.

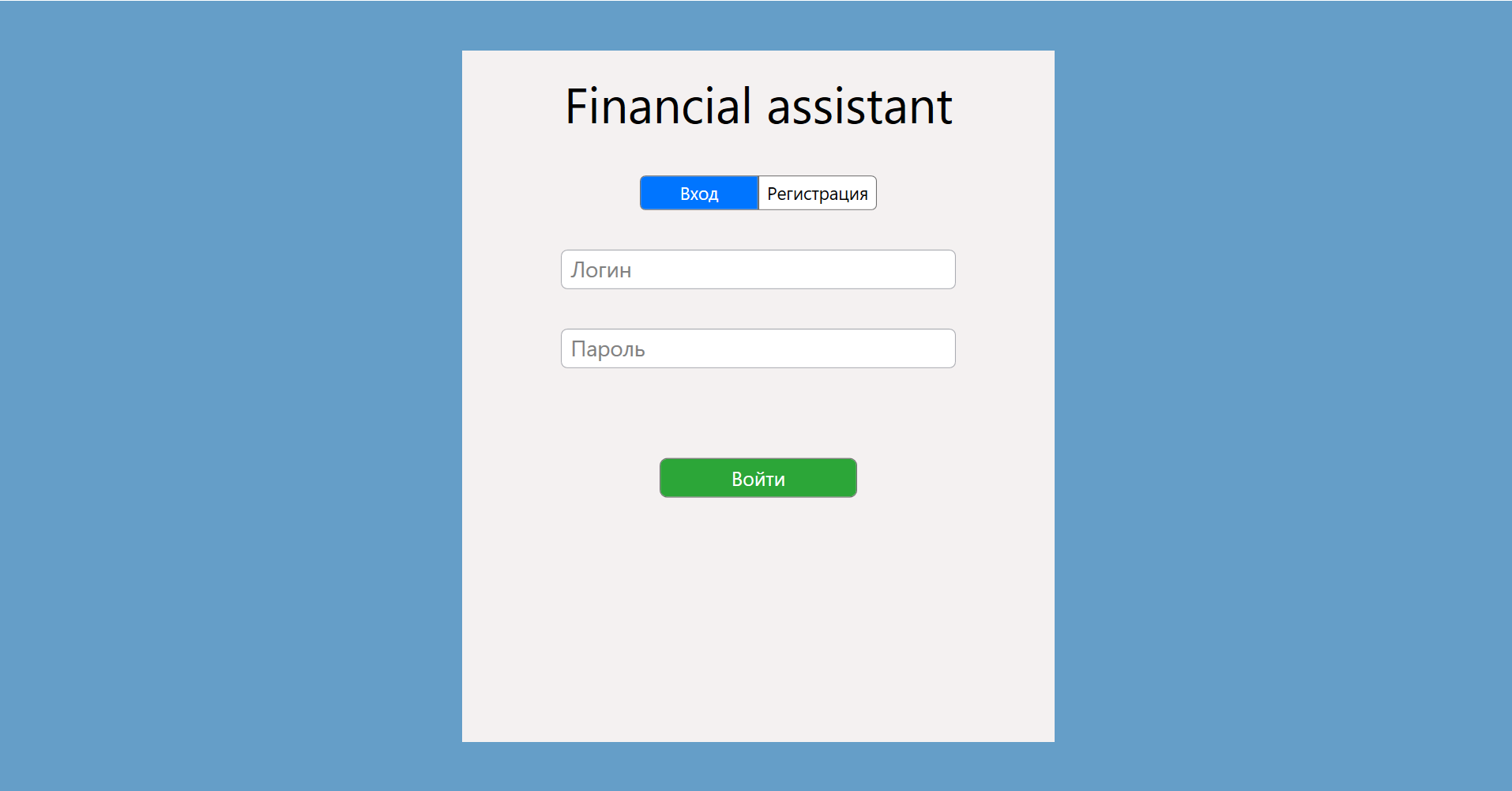


Рисунок 8.1 – Окно авторизации

После авторизации вы попадаете на главное окно, на страницу бюджета. На ней присутствуете график, который отображает, как менялись ваши доходы, расходы и бюджет после каждого добавления сохранения. Снизу под графика есть кнопка “Добавить” в виде плюсика. После нажатия на неё, у вас откроется окно добавления дохода/расхода. Справа от кнопки находится фильтр времени. После выбора там промежутка времени, график отобразит информацию за этот промежуток. Справа сверху отображается общая информация о бюджете: бюджет, сумма доходов, сумма расходов. Правее от информации о бюджете находиться иконка вашего профиля, ниже, имя профиля. Нажав на иконку вашего профиля, у вас откроется окно редактирования своего профиля. Здесь вы сможете поменять фото профиля и имя, а также выйти из аккаунта. Слева, на главной странице, меню навигации по страницам приложения. Скриншот страницы бюджет представлен на рисунке 8.2

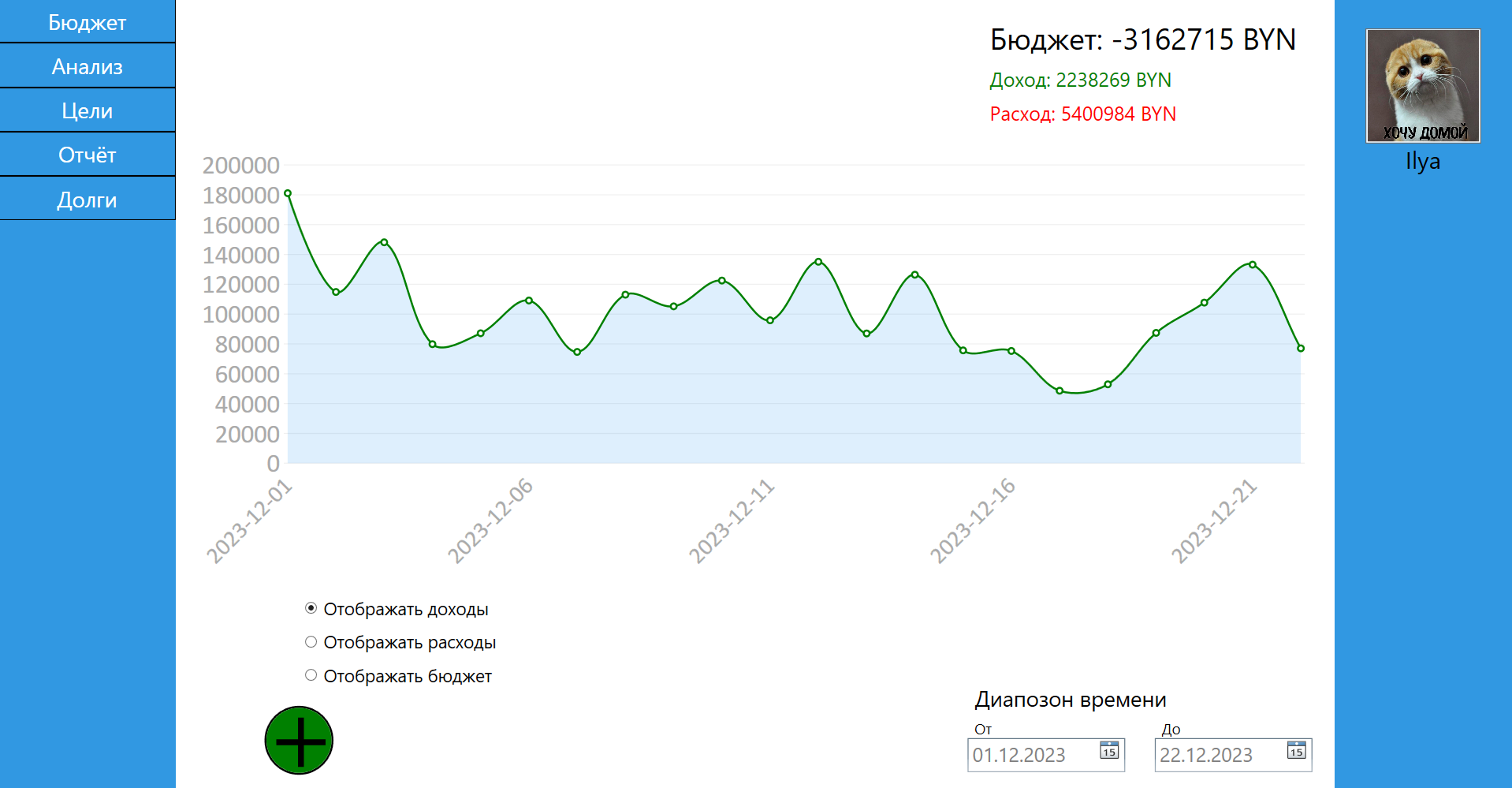


Рисунок 8.2 – Страница «Бюджет»

На странице «Анализ» отображаются диаграммы категорий доходов расходов за выбранный промежуток времени, а также текстовая информация. Скриншот страницы представлен на рисунке 8.3.

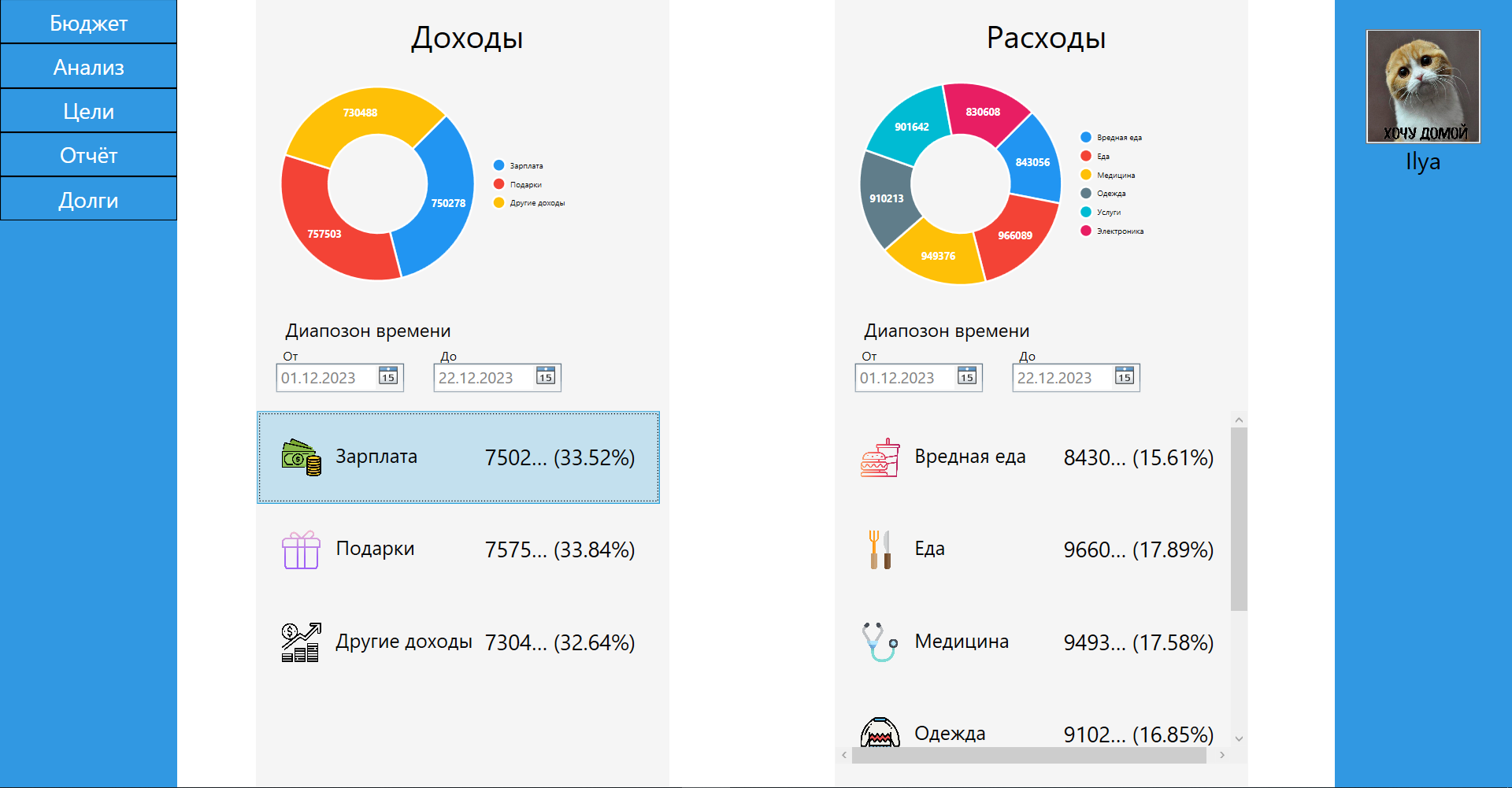


Рисунок 8.3 – Страница «Анализ»

На странице «Цель» вы может добавить себе цель, на которую вы копите, установить на неё цену и изображение. Затем, нажав на кнопку вложить, в виде плюсика, ввести сумму, которую вы отложили на цель. После чего индикатор накопления заполниться. Таким образом на этой странице вы можете следить за процессом накопления. Страница представлена на рисунке 8.4.

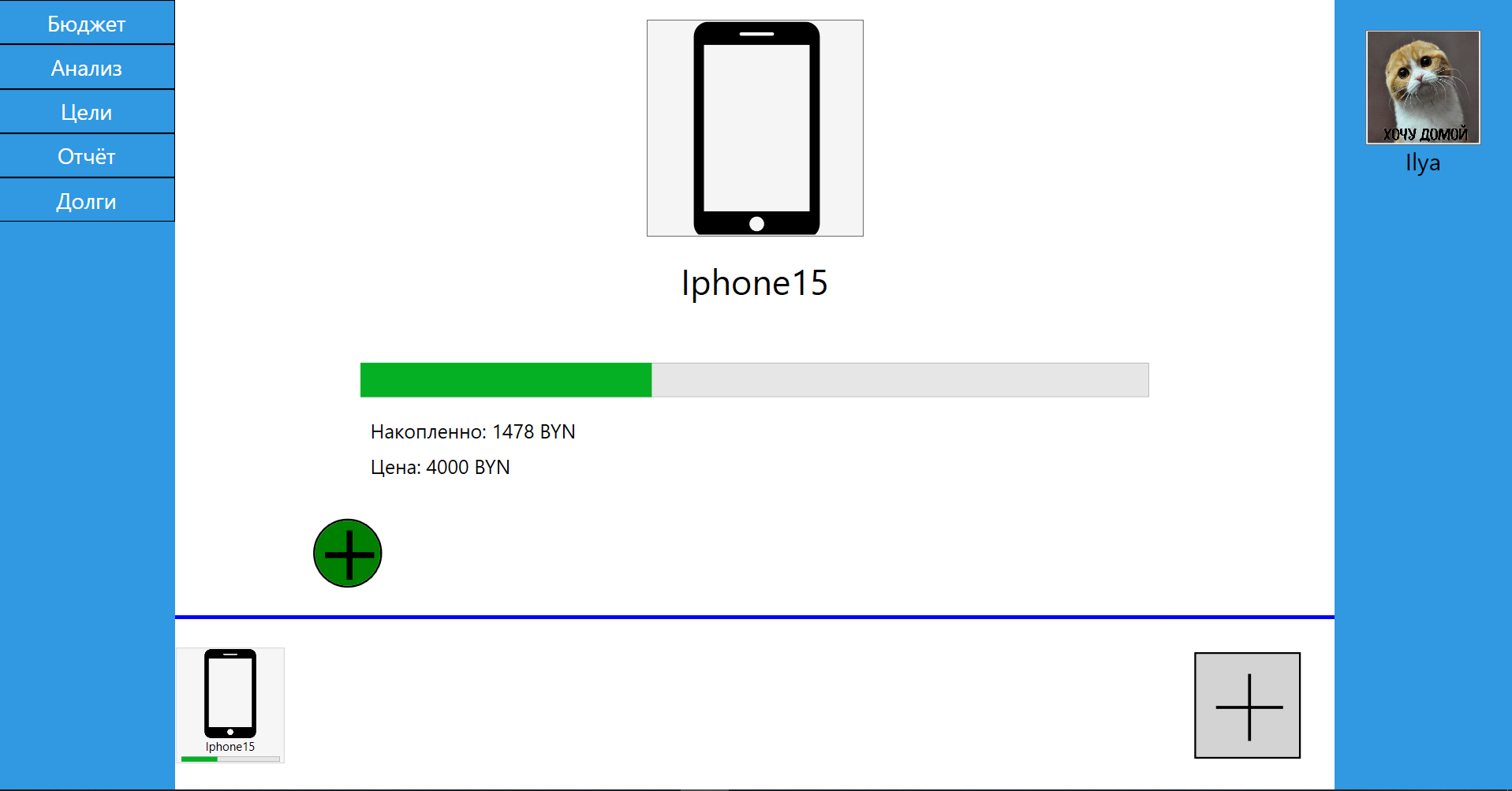


Рисунок 8.4 – Страница «Цели»

На странице «Отчёт» вы можете посмотреть историю ваших сохранений. Страница представлена на рисунке 8.5.

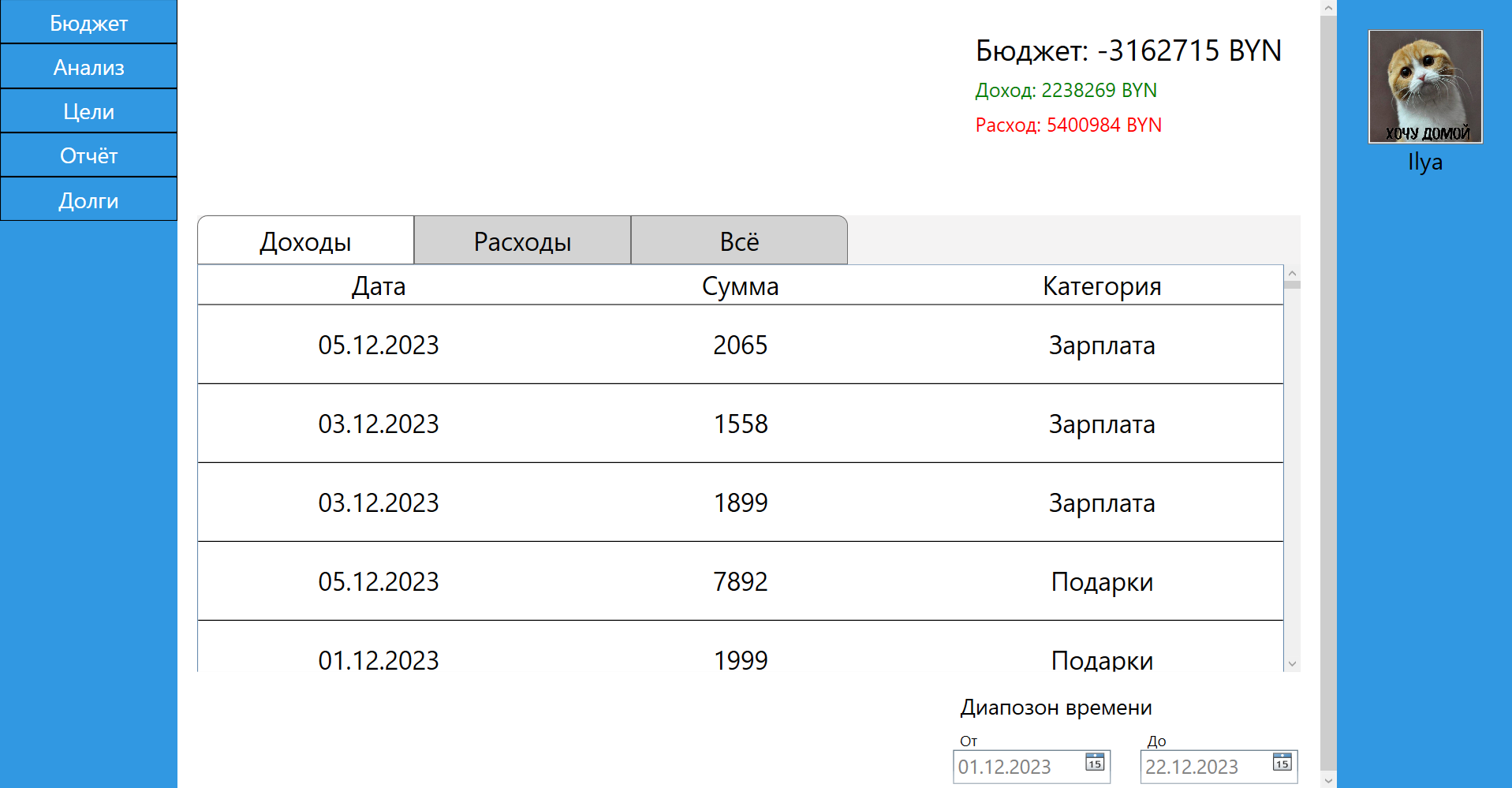


Рисунок 8.5 – Страница «Отчёт»

На странице «Долги» вы можете добавить себе долг, затем, кликнув ПКМ по долгу в таблице, выбрать пункт «оплатить», тем самым сохранить информацию о совершении платежа долга. Сам долг представляет из себя определённое количество платежей, которые надо оплатить. Сумма каждого следующего платежа увеличивается на установленный процент. Скриншот страницы «Долги» представлен рисунке 8.6.

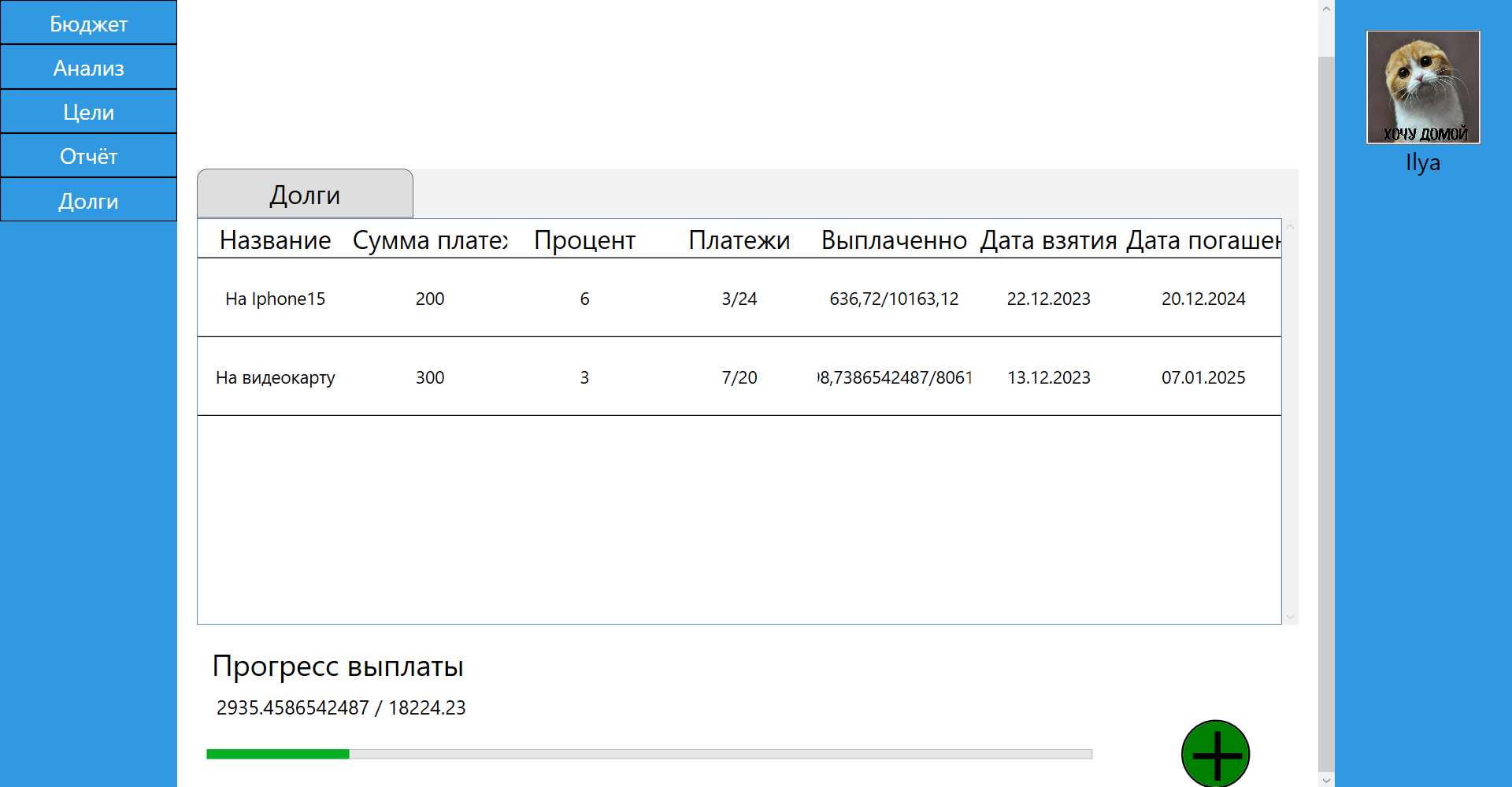


Рисунок 8.7 – Страница «Долги»

Если же вы вошли в аккаунт с ролью «администратор», то у вас откроется окно со страницами «Категории» и «Пользователи». Первоначальной страницей будет «Категории». На ней вы может управлять категориями. Страница представлена на рисунке 8.7

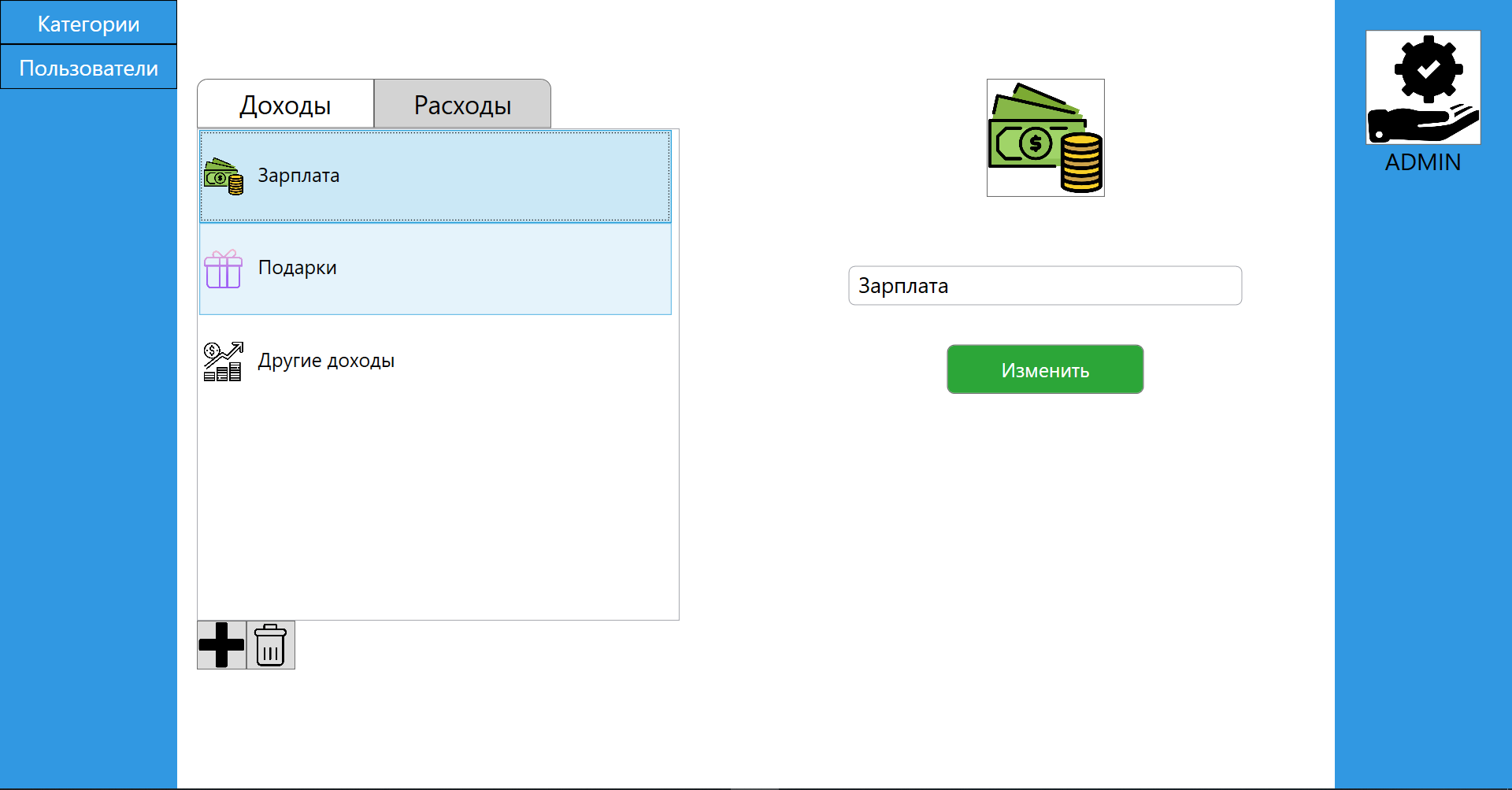


Рисунок 8.7 – Страница «Категории»

На странице «Пользователи», соответственно вы можете управлять учётными записями пользователей. На странице находиться таблица с пользователями. Важной деталью будет то, что пароли отображаются в хешированном виде. Скриншот страницы представлен на рисунке 8.8.

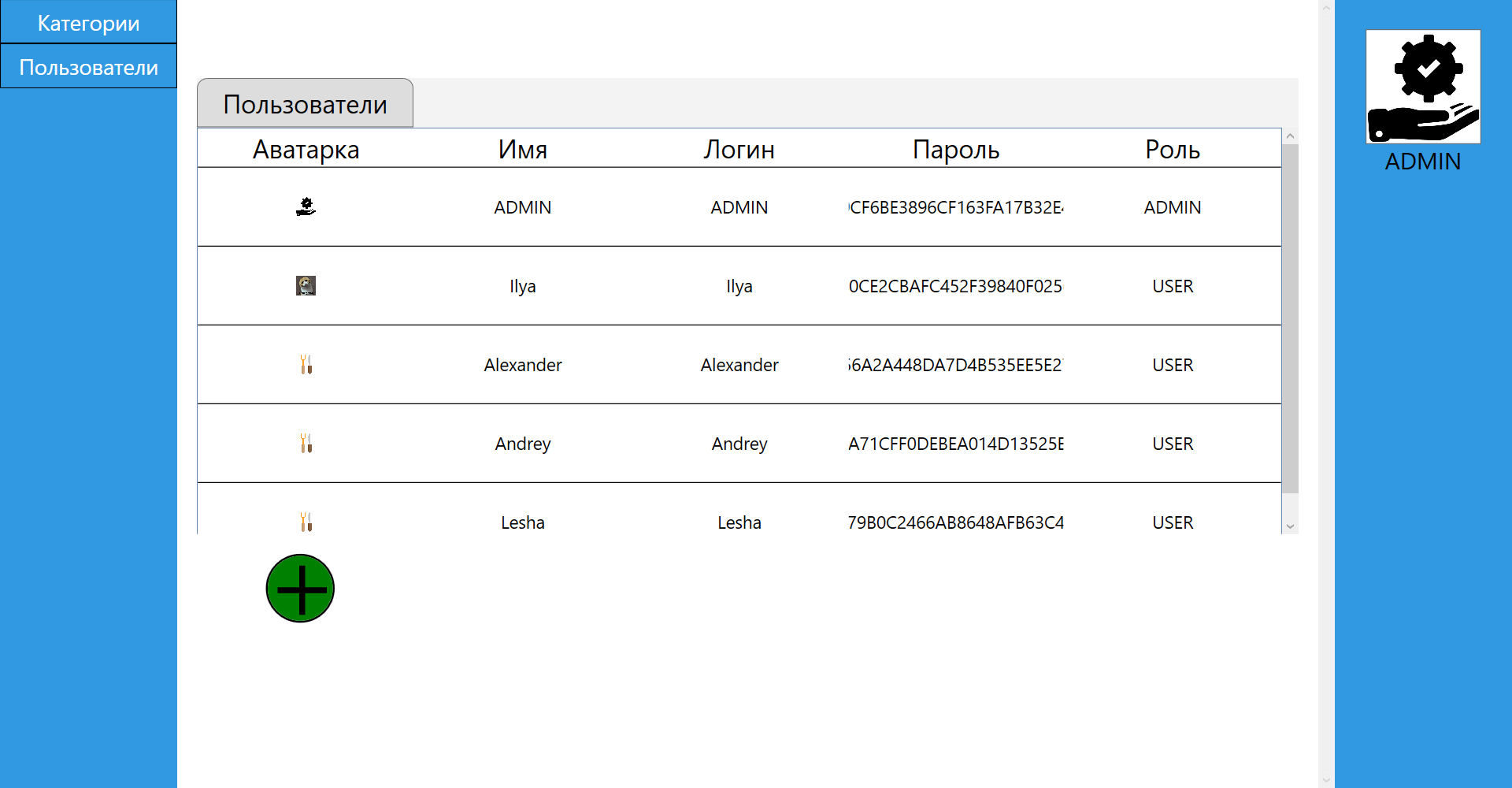


Рисунок 8.7 – Страница «Пользователи»

Как мы видим, приложение представляет из себя окна с несколькими страницами. Большинство данных представлено в виде таблиц или списков. Для взаимодействия большинством элементов в таблицах и списках, необходимо нажать по ним правой кнопкой мыши.

Заключение

Результатом выполнения курсового проекта стало приложение с использованием технологий WPF и ODP для взаимодействия с базой данных. В ходе курсового проекта была использована база данных Oracle, которая спроектирована для хранения данных финансового учёта.

Были реализованы все основные требования, указанные в листе задания вместе со следующими пунктами:

* использование системы управления базами данных (СУБД);
* доступ к данным должен осуществляться только через соответствующие хранимые процедуры.

Возможности администратора:

* управление базой данных категорий;
* управление базой данных пользователей.

Возможности пользователя:

* добавления сохранения (дохода/расхода);
* удаление сохранения (дохода/расхода);
* добавления цели;
* удаление цели;
* редактирование цели;
* сохранение информации о вложении в цель;
* просмотр прогресса достижения целей;
* добавления долга
* удаление долга;
* редактирование долга;
* сохранение информации о оплате долга;
* просмотр прогресса выплаты долгов.

Функциональные требования программного средства:

* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем различных данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* Должен производиться анализ финансового состояния пользователя.

Данный курсовой проект является примером разработки базы данных и приложения для работы с ней. Протестировав все компоненты приложения, можно прийти к заключению, что все основные требования выполнены и приложение работает исправно.

Список используемых источников

1. METANIT.COM Сайт о программировании [Электронный ресурс] Режим доступа: https://metanit.com/sharp/adonet/1.1.php – Дата доступа: 10.12.2023.
2. Oracle-dba.ru [Электронный ресурс] / Режим доступа: <https://oracle-dba.ru> – Дата доступа: 10.12.2023.
3. Cerebrosql Мониторинг работы БД Oracle [Электронный ресурс] / Режим доступа: https://www.cerebrosql.com/metrics-general-window – Дата доступа: 10.12.2023.
4. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm – Дата доступа: 10.12.2023.
5. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 10.12.2023.

Приложение А

CREATE TABLE Users (

ID INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,

Login NVARCHAR2(255) NOT NULL UNIQUE,

Password NVARCHAR2(50) NOT NULL,

Name NVARCHAR2(50) NOT NULL,

Picture BLOB,

Role SMALLINT DEFAULT 0 NOT NULL CHECK (Role IN (0, 1)),

PRIMARY KEY (ID)

) TABLESPACE FA\_TS;

CREATE TABLE Categories (

CategoryID INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,

CategoryName NVARCHAR2(50) NOT NULL,

Type SMALLINT DEFAULT 0 NOT NULL CHECK (Type IN (0, 1)),

Picture BLOB,

PRIMARY KEY (CategoryID)

) TABLESPACE FA\_TS;

CREATE TABLE Saves (

SaveID INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,

Amount NUMBER(18, 2) DEFAULT 0 NOT NULL CHECK (Amount >= 0),

SaveDate DATE NOT NULL,

CategoryID INT NOT NULL,

UserID INT NOT NULL,

PRIMARY KEY (SaveID),

FOREIGN KEY(UserID) REFERENCES Users(ID),

FOREIGN KEY(CategoryID) REFERENCES Categories(CategoryID)

) TABLESPACE FA\_TS;

CREATE TABLE Goals (

GoalID INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,

NameGoal NVARCHAR2(255) NOT NULL,

Accumulated NUMBER(18, 2) DEFAULT 0 NOT NULL CHECK (Accumulated >= 0),

Price NUMBER(18, 2) DEFAULT 0 NOT NULL CHECK (Price >= 0),

Picture BLOB,

UserID INT NOT NULL,

PRIMARY KEY (GoalID),

FOREIGN KEY(UserID) REFERENCES Users(ID)

) TABLESPACE FA\_TS;

CREATE TABLE Debts (

DebtID INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,

NameDebt NVARCHAR2(255) NOT NULL,

Paid INT DEFAULT 0 NOT NULL CHECK (Paid >= 0),

AmountPayment NUMBER(18, 2) DEFAULT 0 NOT NULL CHECK (AmountPayment >= 0),

NumberPayments INT DEFAULT 1 NOT NULL CHECK (NumberPayments > 0),

Percent NUMBER DEFAULT 0 NOT NULL CHECK (Percent >= 0 AND Percent <= 100),

ReceivingDate DATE NOT NULL,

MaturityDate DATE NOT NULL,

UserID INT NOT NULL,

PRIMARY KEY (DebtID),

FOREIGN KEY(UserID) REFERENCES Users(ID)

) TABLESPACE FA\_TS;

Приложение Б

|  |  |  |  |
| --- | --- | --- | --- |
| Название процедуры | Название параметра | Тип параметра | IN / OUT |
| GetAllUsers | userCursor | nvarchar2 | IN |
| name\_a | nvarchar2 | IN |
| surname\_a | nvarchar2 | IN |
| patronymic\_a | nvarchar2 | IN |
| password\_a | nvarchar2 | IN |
| birthday\_a | date | IN |
| phone\_number\_a | nvarchar2 | IN |
| email\_a | nvarchar2 | IN |
| GetAllUsers | userCursor | SYS\_REFCURSOR | OUT |
| DeleteUser | p\_id | INT | IN |
| DeleteSave | p\_saveId | INT | IN |
| DeleteCategory | p\_categoryID | INT | IN |
| DeleteGoal | p\_goalID | INT | IN |
| DeleteDebt | p\_debtID | INT | IN |
| AlterUserByNameAndPicture | p\_id | INT | OUT |
| p\_name | NVARCHAR2 | IN |
| p\_picture | BLOB | IN |
| AddCategory | p\_categoryName | NVARCHAR2 | IN |
| p\_type | SMALLINT | IN |
| p\_picture | BLOB | IN |
| GetUserByID | userCursor | SYS\_REFCURSOR | OUT |
| role\_a | nvarchar2 | IN |
| GetUserByLogin | p\_login | NVARCHAR2 | IN |
| userCursor | SYS\_REFCURSOR | OUT |
| GetCategoriesByType | p\_type | SMALLINT | IN |
| categoriesCursor | SYS\_REFCURSOR | OUT |
| GetAllGoals | p\_userID | INT | IN |
| goalsCursor | SYS\_REFCURSOR | OUT |
| GetAllDebts | p\_userID | INT | IN |
| debtsCursor | SYS\_REFCURSOR | OUT |
| GetDebtByID | p\_debtID | INT | IN |
| debtsCursor | SYS\_REFCURSOR | OUT |

|  |  |  |  |
| --- | --- | --- | --- |
| CalculateTotalDebts | p\_userID | INT | IN |
| totaDebts | SYS\_ NUMBER | OUT |
| CalculateTotalDebt | p\_userID | INT | IN |
| totaDebts | NUMBER | OUT |
| CalculatePaidDebt | p\_debtID | INT | IN |
| totalDebt | NUMBER | OUT |
| CalculatePaidDebts | p\_userID | INT | IN |
| totaDebts | NUMBER | OUT |
| GetAllSaves | startDate | DATE | IN |
| endDate | DATE | IN |
| p\_userID | INT | IN |
| savesCursor | SYS\_REFCURSOR | OUT |
| AddSaves | p\_amount | NUMBER | IN |
| p\_saveDate | DATE | IN |
| p\_categoryID | INT | IN |
| p\_userID | INT | IN |
| AlterCategory | p\_categoryID | INT | IN |
| p\_categoryName | NVARCHAR2 | IN |
| p\_type | SMALLINT | IN |
| p\_picture | BLOB | IN |
| CalculateBudget | startDate | DATE | IN |
| endDate | DATE | IN |
| p\_userID | INT | IN |
| budget | NUMBER | OUT |
| AddUser | p\_login | NVARCHAR2 | IN |
| p\_password | NVARCHAR2 | IN |
| p\_name | NVARCHAR2 | IN |
| p\_picture | BLOB | IN |
| p\_role | SMALLINT | IN |
| AlterUserWithoutPassword | p\_id | INT | IN |
| p\_name | NVARCHAR2 | IN |
| p\_login | NVARCHAR2 | IN |
| p\_picture | BLOB | IN |
| p\_role | SMALLINT | IN |

|  |  |  |  |
| --- | --- | --- | --- |
| GetSavesByCategoryType | startDate | DATE | IN |
| endDate | DATE | IN |
| p\_type | SMALLINT | IN |
| p\_userID | INT | IN |
| savesCursor | SYS\_REFCURSOR | OUT |
| AddGoal | p\_nameGoal | NVARCHAR2 | IN |
| p\_accumulated | NUMBER | IN |
| p\_price | NUMBER | IN |
| p\_picture | BLOB | IN |
| p\_userID | INT | IN |
| AlterGoal | p\_goalID | INT | IN |
| p\_nameGoal | NVARCHAR2 | IN |
| p\_accumulated | NUMBER | IN |
| p\_price | NUMBER | IN |
| p\_picture | BLOB | IN |
| CalculateSaves | startDate | DATE | IN |
| endDate | DATE | IN |
| p\_userID | INT | IN |
| p\_type | SMALLINT | IN |
| sumSaves | NUMBER | OUT |
| AlterUser | p\_id | INT | IN |
| p\_name | NVARCHAR2 | IN |
| p\_login | NVARCHAR2 | IN |
| p\_password | NVARCHAR2 | IN |
| p\_picture | BLOB | IN |
| p\_role | SMALLINT | IN |
| AddDebt | p\_nameDebt | NVARCHAR2 | IN |
| p\_amountPayment | NUMBER | IN |
| p\_numberPayments | NUMBER | IN |
| p\_percent | NUMBER | IN |
| p\_receivingDate | DATE | IN |
| p\_maturityDate | DATE | IN |
| p\_userID | INT | IN |

|  |  |  |  |
| --- | --- | --- | --- |
| AlterDebt | p\_debtID | INT | IN |
| p\_nameDebt | NVARCHAR2 | IN |
| p\_paid | INT | IN |
| p\_amountPayment | NUMBER | IN |
| p\_numberPayments | INT | IN |
| p\_percent | NUMBER | IN |
| p\_receivingDate | DATE | IN |
| p\_maturityDate | DATE | IN |
| CalculateSavesByCategory | startDate | DATE | IN |
| endDate | DATE | IN |
| p\_categoryID | INT | IN |
| p\_userID | INT | IN |
| p\_type | SMALLINT | IN |
| sumSavesOfCategory | NUMBER | OUT |
| partSaves | NUMBER | OUT |