МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВАЯ РАБОТА

по дисциплине: Технологии разработки программного обеспечения
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Система управления личными финансами

Студент: Дицель Никита Сергеевич
Группа: ИКБО-11-23
Руководитель: Супрун Антон Павлович

Оценка по итогам защиты: _____

Подпись студента:_____ Подпись руководителя: _____

М. РТУ МИРЭА. 2025

Оглавление

Введение

В условиях роста финансовой грамотности населения всё большее внимание уделяется контролю за личными доходами и расходами. Эффективное управление личным бюджетом позволяет не только избежать незапланированных долгов, но и формировать сберегательные привычки и повышать качество жизни. В связи с этим разработка программных инструментов, упрощающих учёт и анализ личных финансов, становится актуальной задачей как для частных пользователей, так и для разработчиков программного обеспечения.

Целью данной курсовой работы является проектирование, разработка и документирование веб-приложения «Система управления личными финансами», предназначенного для автоматизации учёта доходов и расходов, классификации операций по категориям, планирования ежемесячных бюджетов и формирования наглядной отчётности. В ходе работы реализованы ключевые функции, соответствующие принципам безопасного, удобного и гибкого программного обеспечения: многопользовательский режим с изоляцией данных, поддержка импорта и экспорта в формате CSV, визуализация финансовой информации, а также расширяемая архитектура на основе фреймворка Django.

Актуальность проекта обусловлена необходимостью предоставить пользователю автономное, приватное и простое в использовании решение, не зависящее от облачных сервисов и сторонних подписок. При разработке особое внимание уделено соблюдению принципов модульности, тестируемости и безопасности, что обеспечивает надёжность приложения и возможность его дальнейшего развития.

Курсовая работа включает в себя полный цикл разработки программного продукта: от анализа требований и проектирования архитектуры до

реализации, тестирования и составления сопроводительной документации, включая руководство пользователя и инструкции по установке.

1. Общее описание программы

1.1. Назначение программы

Программа «Система управления личными финансами» предназначена для автоматизации учёта личных доходов и расходов, планирования ежемесячных бюджетов, анализа финансовой активности и формирования отчётности. Целью разработки является предоставление пользователю удобного инструмента для повышения финансовой грамотности, контроля за расходами и принятия обоснованных решений в области личного бюджета.

Программа решает следующие задачи:

1. Регистрация и хранение информации о финансовых операциях (доходах и расходах);

2. Классификация операций по пользовательским категориям;

3. Установка и контроль соблюдения ежемесячных бюджетов по категориям расходов;

4. Визуализация данных в виде диаграмм и таблиц;

5. Импорт и экспорт данных в формате CSV для резервного копирования и обмена;

6. Обеспечение изоляции данных между разными пользователями.

1.2. Область применения

Программа ориентирована на частных пользователей, желающих вести учёт личных финансов без использования сторонних облачных сервисов. Она может использоваться как:

- самостоятельное решение для домашней бухгалтерии;
- основа для дальнейшего развития в полноценное веб-приложение с расширенной функциональностью.

1.3. Основные функции

Программа предоставляет следующие функциональные возможности:

- Управление категориями операций
- Пользователь может создавать собственные категории доходов и расходов (например, «Зарплата», «Еда», «Транспорт»).
- Регистрация транзакций
- Добавление, просмотр, фильтрация и удаление финансовых операций с указанием суммы, категории, даты и описания.
- Планирование бюджета
- Установка лимитов расходов по категориям на каждый месяц с возможностью отслеживания фактического выполнения.
- Финансовая отчётность
- Автоматическое формирование сводок по доходам и расходам за текущий месяц, визуализация структуры расходов в виде круговой диаграммы.
- Импорт/экспорт данных
- Поддержка экспорта всех транзакций в CSV-файл и обратного импорта данных из CSV для переноса или резервного копирования.
- Многопользовательский режим

– Каждый зарегистрированный пользователь работает только со своими данными; данные изолированы на уровне базы данных.

1.4. Принципы проектирования

При разработке программы соблюдались следующие принципы:

– Модульность: логически связанные компоненты выделены в отдельные модули (модели, формы, представления, сервисы).
– Гибкость: архитектура позволяет легко расширять функционал (например, добавить поддержку валют или уведомлений).
– Тестируемость: бизнес-логика вынесена в отдельный слой (services.py), что упрощает написание unit-тестов.
– Безопасность: все операции с данными привязаны к авторизованному пользователю; удаление требует подтверждения.
– Удобство использования: интуитивно понятный веб-интерфейс с поддержкой фильтрации и визуализации.
– Программа полностью соответствует требованиям курсовой работы: объём исходного кода превышает 1000 строк, код снабжён комментариями, реализована гибкая архитектура, подготовлена полная документация.

2. Архитектура программы

2.1. Общая архитектурная модель

Программа реализована на основе веб-фреймворка Django и использует архитектурный паттерн MVT (Model-View-Template) — адаптацию классического MVC (Model-View-Controller), принятую в экосистеме Django:

Model (Модель) — отвечает за работу с данными и бизнес-логику на уровне объектов. Реализована в файлах core/models.py. Включает сущности: Transaction (транзакция), Category (категория), Budget (бюджет).

View (Представление) — обрабатывает HTTP-запросы, координирует взаимодействие между моделью и шаблоном, возвращает HTTP-ответ. Реализована в core/views.py с использованием функциональных представлений и декоратора @login_required.

Template (Шаблон) — отвечает за отображение данных в виде HTML. Шаблоны хранятся в core/templates/core/ и используют наследование от базового шаблона base.html.

Такой подход обеспечивает чёткое разделение ответственности, упрощает сопровождение и тестирование кода.

2.2. Модульная структура проекта

Проект разделён на логически независимые приложения и модули:

core — основное приложение, содержащее всю бизнес-логику:

models.py — определение моделей данных;

views.py — обработка запросов;

forms.py — валидация и отображение форм;

services.py — слой бизнес-логики (агрегации, импорт/экспорт);

urls.py — маршрутизация;

templates/ — HTML-шаблоны;

tests.py — автоматизированные тесты.

accounts — управление пользователями (авторизация, регистрация через встроенные механизмы Django).

personal_finance/ — конфигурационный пакет проекта (настройки, главные URL).

Такая структура соответствует принципу DRY (Don’t Repeat Yourself) и позволяет легко расширять функционал (например, добавить приложение api для мобильного клиента).

2.3. Использованные паттерны проектирования

В разработке применены следующие паттерны проектирования:

Service Layer

Бизнес-логика (расчёт сводок, импорт/экспорт CSV, сравнение с бюджетом) вынесена в отдельный модуль services.py. Это изолирует логику от представлений, упрощает модульное тестирование, позволяет повторно использовать функции в разных частях приложения.

Front Controller

Все HTTP-запросы обрабатываются централизованно через механизм маршрутизации Django (urls.py), что обеспечивает единый входной поток управления.

Template Method

Используется во встроенных механизмах Django (например, валидация форм через clean_<field>()), что позволяет гибко расширять поведение без изменения базового кода.

Dependency Injection

Зависимости (например, текущий пользователь) передаются явно в формы и сервисы, что повышает тестируемость компонентов.

2.4. Гибкость и расширяемость

Архитектура программы спроектирована с учётом будущего развития:

- Добавление новых типов операций — достаточно расширить модель Category.
- Поддержка валют — можно добавить поле currency в Transaction и логику конвертации в services.py.
- Мобильный API — создание нового приложения api с использованием Django REST Framework.
- Уведомления — интеграция с Celery или email-рассылкой через сигналы Django.
- Аналитика — подключение библиотек вроде Pandas для прогнозирования бюджета.

Все изменения могут быть внесены без переписывания существующей логики благодаря чёткому разделению слоёв и соблюдению принципов SOLID.

2.5. Безопасность

Все данные привязаны к авторизованному пользователю (user=request.user). Удаление транзакций требует подтверждения (HTTP POST + JavaScript-диалог). Используется CSRF-защита для всех форм. Пароли хранятся в хэшированном виде (стандартный механизм Django).

Таким образом, архитектура программы соответствует современным требованиям к веб-приложениям: она модульна, тестируема, безопасна и легко расширяема.

3. Системные требования к программе

3.1. Общие положения

Системные требования определяют минимальные и рекомендуемые характеристики программного и аппаратного обеспечения, необходимые для корректной установки, запуска и эксплуатации программы «Система управления личными финансами». Требования сформулированы в соответствии с принципами переносимости, совместимости и минимальной ресурсоёмкости, что обеспечивает возможность использования программы на широком спектре устройств.

3.2. Требования к программному обеспечению

3.2.1. Серверная часть (бэкенд)

Для запуска серверной части приложения необходимы следующие компоненты:

Компонент	Минимальная версия	Рекомендуемая версия	Назначение
Операционная система	Любая, поддерживающая Python 3.8+	Windows 10/11, Ubuntu 20.04+, macOS Monterey+	Выполнение приложения
Python	3.8	3.10-3.12	Интерпретатор языка программирования
Django	4.2	5.0-5.2	Веб-фреймворк
Python-dateutil	2.8	2.9+	Работа с датами и временем
pip	21.0	Последняя стабильная	Установка зависимостей

Примечание: Программа не требует установки отдельной СУБД — по умолчанию используется встроенная база данных SQLite3, входящая в стандартную библиотеку Python. При необходимости возможна миграция на PostgreSQL или MySQL без изменения кода приложения.

3.2.2. Клиентская часть (фронтенд)

Для взаимодействия с программой через веб-интерфейс требуется:

Компонент	Минимальная версия	Рекомендуемая версия	Назначение
Веб-браузер	Любой с поддержкой HTML5 и CSS3	Google Chrome 100+, Mozilla Firefox 100+, Microsoft Edge 100+, Safari 15+	Отображение интерфейса
JavaScript	ES5	ES6	Работа интерактивных элементов
Интернет-соединение	Не требуется	Не требуется	Программа работает полностью в оффлайн-режиме

Примечание: Для полной функциональности (отображение диаграмм в разделе «Отчёты») рекомендуется наличие доступа к CDN, так как библиотека Chart.js подключается через публичный URL. Однако при отсутствии интернета программа сохраняет базовую работоспособность: учёт транзакций, категории, экспорт/импорт CSV.

3.3. Требования к аппаратному обеспечению

Программа является легковесным веб-приложением и не предъявляет высоких требований к аппаратным ресурсам.

Ресурс	Минимальные требования	Рекомендуемые требования
Процессов	1 ядро, 1 ГГц	2+ ядра, 2 ГГц
Оперативная память	512 МБ	2 ГБ
Место на диске	50 МБ	100 МБ
Сетевой адаптер	Не требуется	Не требуется

Примечание: Требования указаны для локального запуска через встроенный сервер разработки Django (runserver). При развёртывании в production-среде (например, через Gunicorn + Nginx) требования могут быть скорректированы в зависимости от нагрузки.

3.4. Требования к окружению разработки

Для модификации и расширения программы рекомендуется использовать следующее окружение:

- Текстовый редактор / IDE: VS Code, PyCharm, Sublime Text с поддержкой Python и Django.
- Виртуальное окружение: venv или virtualenv для изоляции зависимостей.
- Система контроля версий: Git (рекомендуется для отслеживания изменений и подготовки курсовой).
- Терминал / командная строка: для выполнения команд Django (migrate, runserver, test).

3.5. Совместимость и ограничения

Программа не поддерживает одновременную работу нескольких пользователей через один аккаунт — каждый пользователь должен иметь собственную учётную запись. Импорт CSV ожидает строгий формат..

Удаление категории невозможно, если по ней есть транзакции (ограничение on_delete=models.PROTECT в модели).

3.6. Зависимости проекта

Полный список зависимостей фиксируется в файле requirements.txt. Установка выполняется командой: pip install -r requirements.txt. Это гарантирует воспроизводимость окружения на любой машине.

Таким образом, система спроектирована с учётом минимальных требований к ресурсам, что делает её доступной для использования на большинстве современных персональных компьютеров и ноутбуков без необходимости в дополнительных лицензиях или дорогостоящем оборудовании.

4. Инструкция по сборке и установке программы

4.1. Общие сведения

Программа может быть установлена и запущена на любом компьютере с операционной системой Windows, Linux или macOS. Процесс состоит из двух этапов:

На машине разработчика — выполняются действия по подготовке и тестированию программы (этот этап уже завершён при сдаче курсовой).

На целевой машине — пользователь устанавливает и запускает готовую программу.

Ниже указано, какие шаги относятся к каждому этапу.

4.2. Действия на машине разработчика

Эти шаги выполнял разработчик:

Создание виртуального окружения

Для изоляции зависимостей была создана папка venv с помощью команды:

python -m venv venv

Установка зависимостей

Все необходимые библиотеки (Django, python-dateutil) были установлены в виртуальное окружение из файла requirements.txt.

Проверка работоспособности

Программа была протестирована: запущена локально, проверены все функции (добавление транзакций, экспорт, отчёты и т.д.).

4.3. Действия на целевой машине

Эти шаги должен выполнить любой человек, который хочет запустить программу на своём компьютере.

4.3.1. Подготовка окружения

Убедитесь, что на компьютере установлен Python версии 3.8 или выше.

Проверить версию можно командой в терминале или командной строке:

python --version

Если Python не установлен — скачайте его с официального сайта: https://www.python.org/downloads/

Создайте виртуальное окружение (это изолированная среда для программы):

Откройте терминал в папке с проектом и выполните:

python -m venv venv

Активируйте виртуальное окружение:

В Windows:

venv\Scripts\activate

В Linux или macOS:

source venv/bin/activate

4.3.2. Установка зависимостей

Убедитесь, что вы находитесь в папке проекта (где лежит файл manage.py).

Выполните команду для установки всех необходимых библиотек:

pip install -r requirements.txt

4.3.3. Инициализация базы данных

Выполните миграции, чтобы создать структуру базы данных:

python manage.py migrate

Создайте учётную запись администратора (потребуется для входа в программу):

python manage.py createsuperuser

Следуйте инструкциям: введите имя пользователя, email (можно пропустить) и пароль.

4.3.4. Запуск программы

Запустите встроенный веб-сервер:

python manage.py runserver

Откройте в браузере адрес:

http://127.0.0.1:8000/accounts/login/

Войдите, используя имя пользователя и пароль, созданные на шаге 4.3.3.

4.4. Структура проекта (на целевой машине)

После установки в папке проекта будут следующие ключевые файлы и папки:

manage.py — основной файл для запуска команд Django

personal_finance/ — настройки проекта

core/ — основная логика программы (модели, формы, шаблоны)

static/ — файлы оформления (стили, скрипты)

requirements.txt — список необходимых библиотек

db.sqlite3 — создаётся автоматически после команды migrate (хранит все ваши данные)

4.5. Возможные проблемы и решения

Ошибка "python не является командой" → Python не установлен или не добавлен в PATH. Установите Python с опцией "Add to PATH".

Не отображаются стили или диаграммы → проверьте подключение к интернету (Bootstrap и Chart.js загружаются из облака).

После входа перенаправляет на ошибку 404 → убедитесь, что в файле personal_finance/settings.py есть строка:

LOGIN_REDIRECT_URL = '/'

5. Методика проверки программы

5.1. Общие положения

Проверка корректности работы программы осуществляется комбинированным методом, включающим автоматизированное модульное тестирование и ручное функциональное тестирование. Такой подход обеспечивает высокую степень покрытия функциональности и позволяет выявить как логические ошибки в коде, так и проблемы взаимодействия с пользователем.

5.2. Автоматизированное тестирование

Автоматизированные тесты реализованы с использованием встроенного фреймворка unittest, интегрированного в Django. Все тесты расположены в файле core/tests.py и охватывают следующие компоненты:

Модели данных: проверка корректности создания объектов Category, Transaction, Budget; соблюдение ограничений (например, уникальность пары «название + тип + пользователь» для категории).

Формы: валидация входных данных (положительная сумма, корректная дата, привязка к пользователю), обработка ошибок.

Сервисный слой: корректность агрегационных функций (расчёт месячного дохода/расхода), импорт и экспорт CSV.

Представления: проверка доступности страниц, корректность HTTP-статусов, обработка POST-запросов.

Запуск тестов осуществляется командой:

python manage.py test

Ожидаемый результат: все тесты проходят успешно (статус OK), количество обнаруженных ошибок и сбоев — 0.

Примеры проверяемых сценариев:

1. Создание транзакции с отрицательной суммой должно вызывать ошибку валидации.

2. Экспорт в CSV должен содержать заголовок и все транзакции пользователя.

3. Импорт CSV с двумя строками данных должен создать ровно две новые транзакции.

4. Удаление транзакции должно быть невозможно для неавторизованного пользователя или при попытке удалить чужую запись.

```
PS C:\Users\redmi\Documents\kursach_5\personal_finance> python manage.py test
Found 16 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
................
----------------------------------------------------------------------
Ran 16 tests in 19.538s

OK
```

Рисунок 1 – Результат автоматизированного тестирования

5.3. Ручное функциональное тестирование

Ручное тестирование проводится путём последовательного выполнения пользовательских сценариев в веб-интерфейсе. Ниже приведены ключевые тест-кейсы.

Тест-кейс 1: Регистрация и вход в систему

1. Перейти на страницу /accounts/login/.

2. Ввести корректные учётные данные.

3. Убедиться, что происходит перенаправление на главную страницу (/).

4. Проверить наличие элементов навигации (кнопки «Транзакции», «+ Транзакция» и др.).

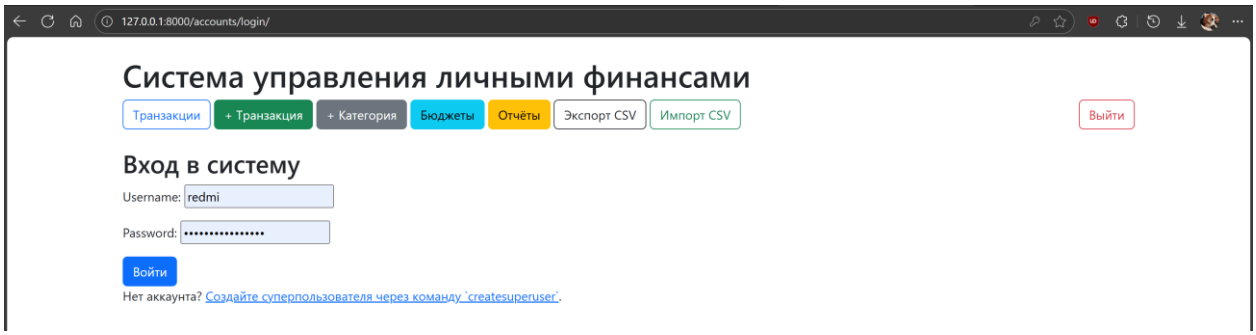

Рисунок 2 – Переходим на страницу /accounts/login

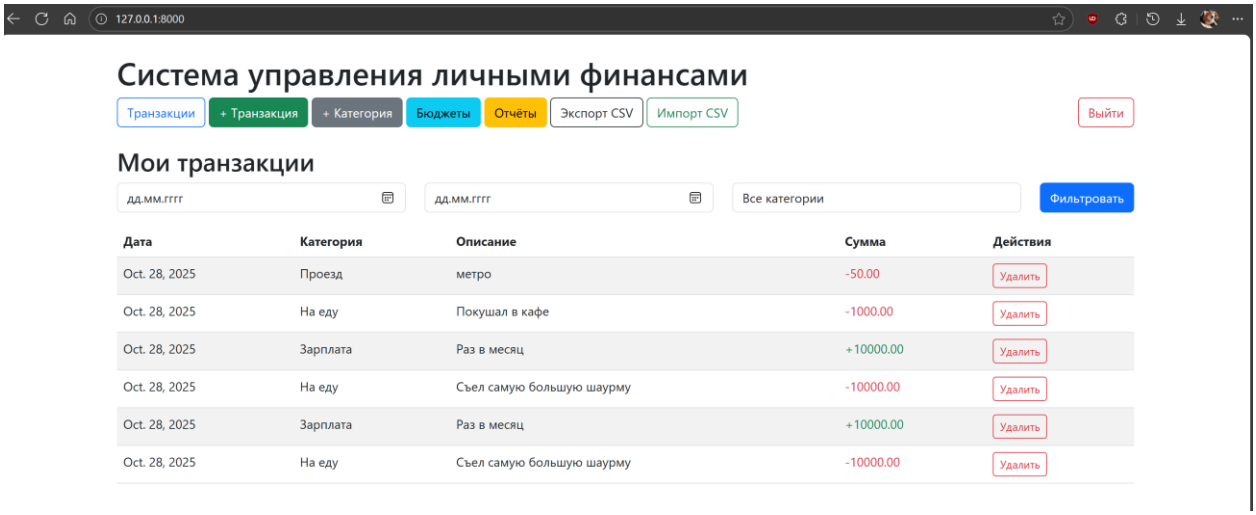

Рисунок 3 – Переходим на главную страницу

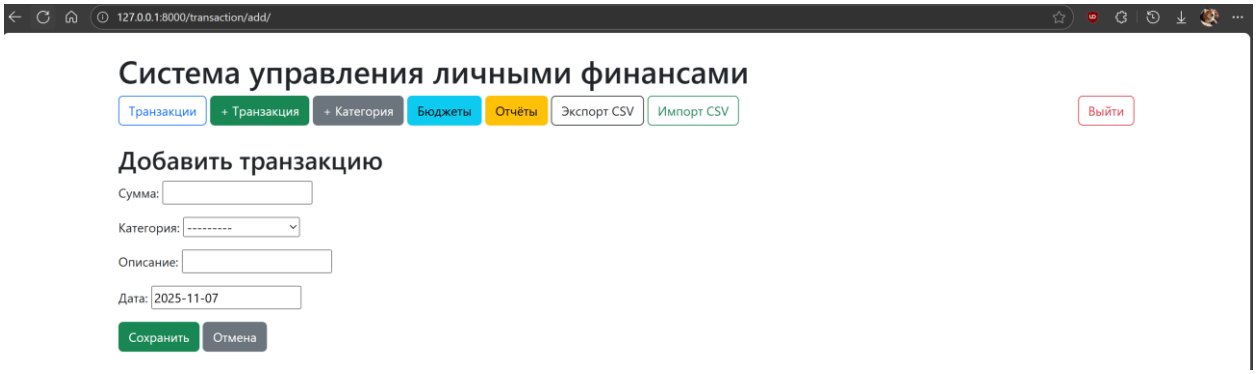

Рисунок 4 – Кнопки «Транзакция» и «+Транзакция» работают

Тест-кейс 2: Управление категориями

1. Нажать кнопку «+ Категория».
2. Заполнить форму: название — «Продукты», тип — «Расход».
3. Сохранить.

4. Убедиться, что категория появилась в списке при создании транзакции.

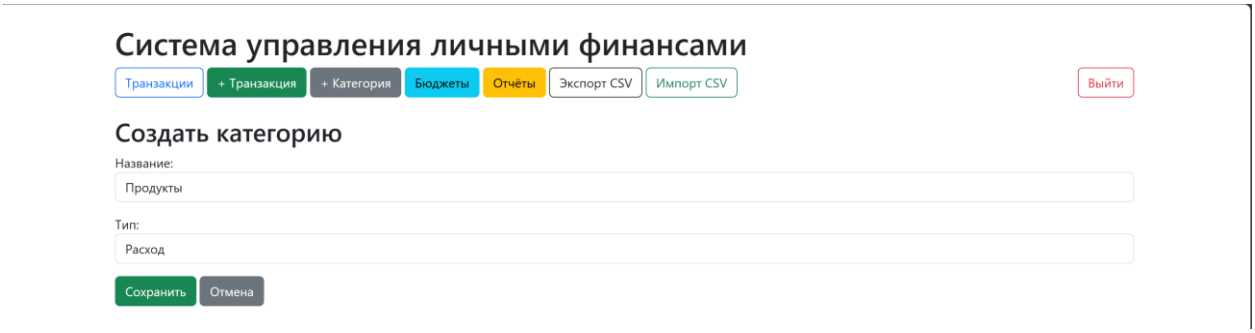

Рисунок 5 – Добавляем категорию

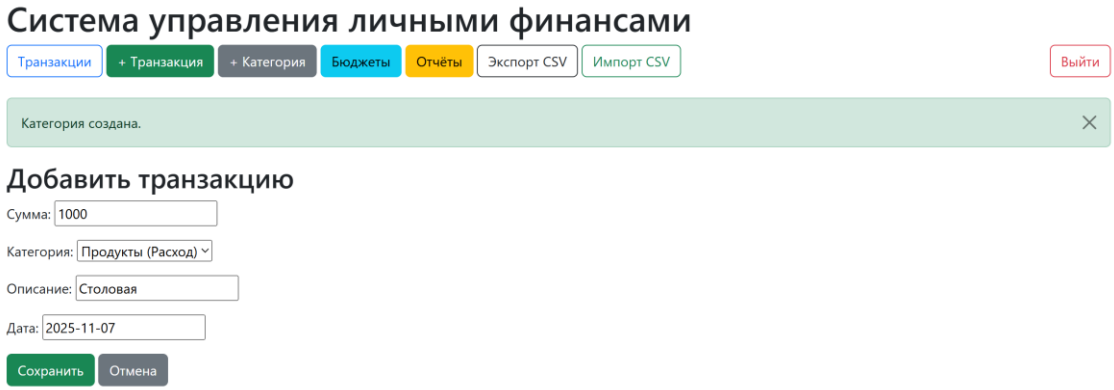

Рисунок 6 – Категория успешно добавлена

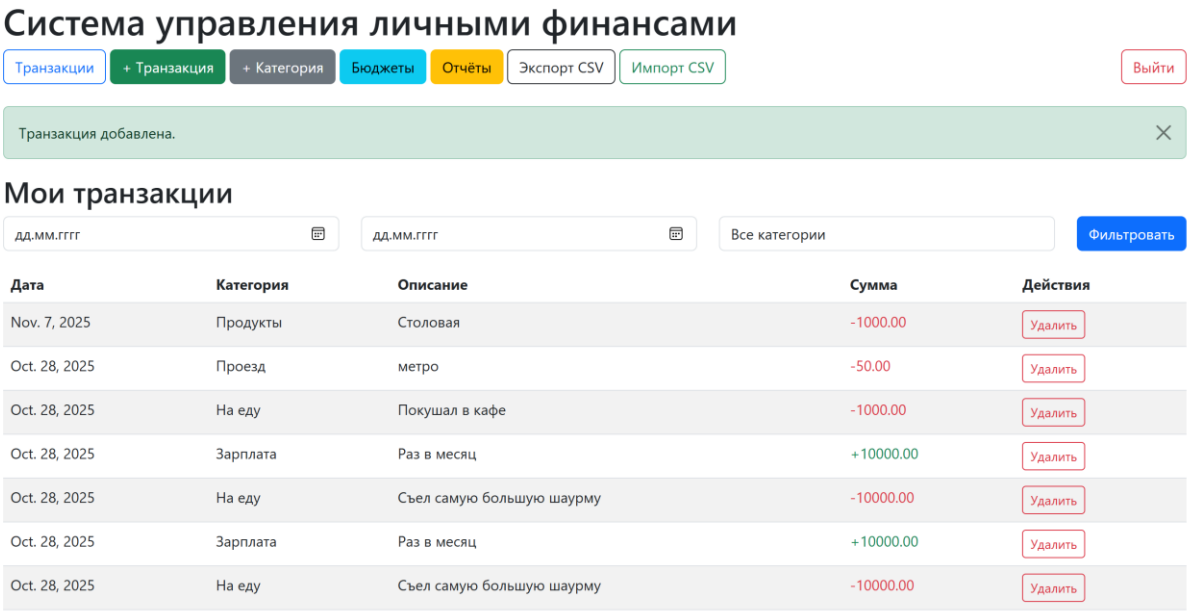

Рисунок 7 – Транзакция успешно добавлена с новой категорией

Тест-кейс 3: Добавление и удаление транзакции

1. Перейти на страницу «+ Транзакция».

2. Выбрать категорию «Продукты», сумму 1500, дату сегодня.

3. Сохранить.

4. Убедиться, что транзакция отображается в списке.

5. Нажать «Удалить» → подтвердить действие.

6. Убедиться, что транзакция исчезла из списка.

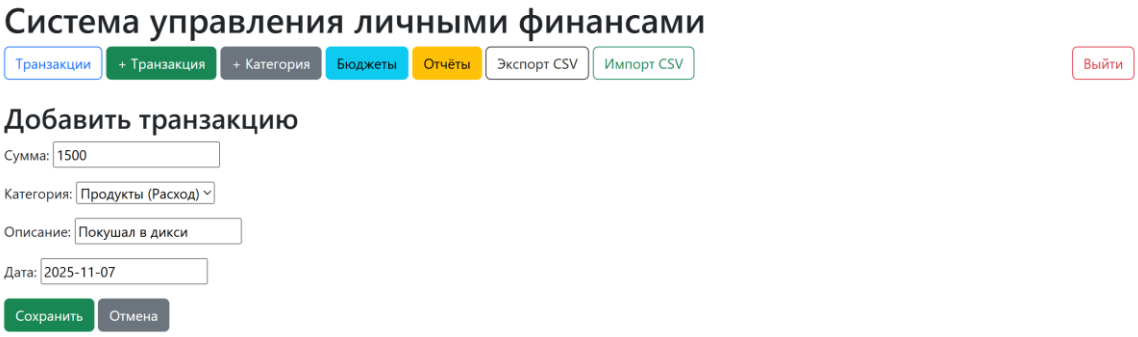

Рисунок 8 – Добавлена новая транзакция

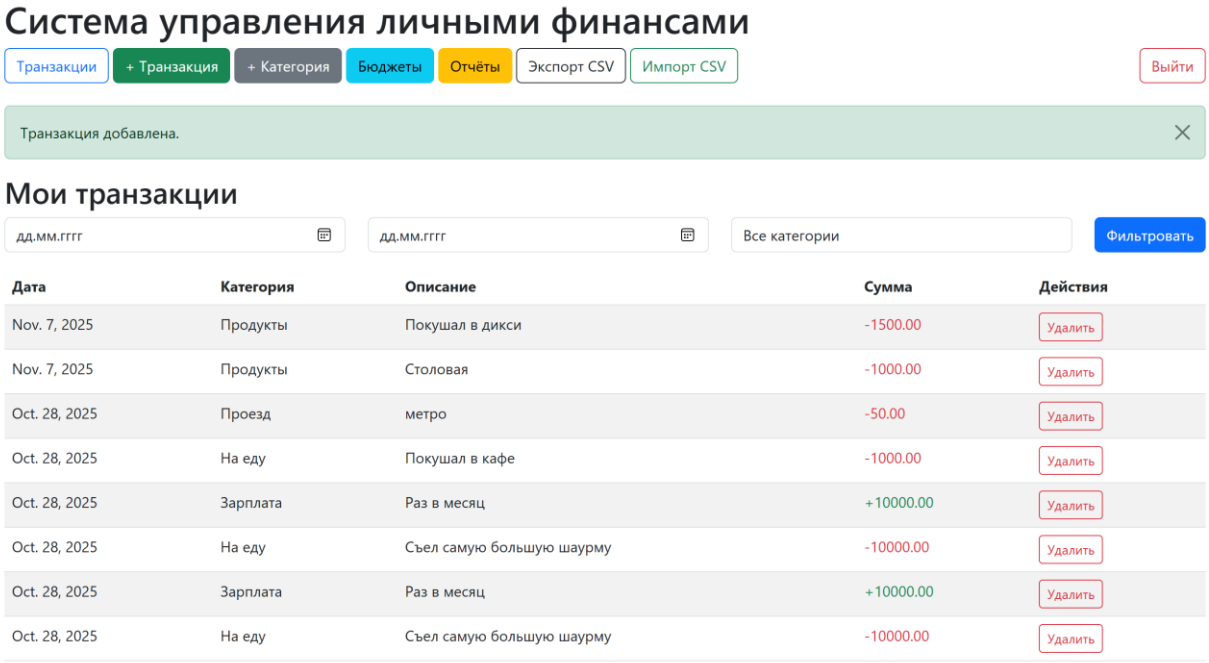

Рисунок 9 – Транзакция успешно добавлена

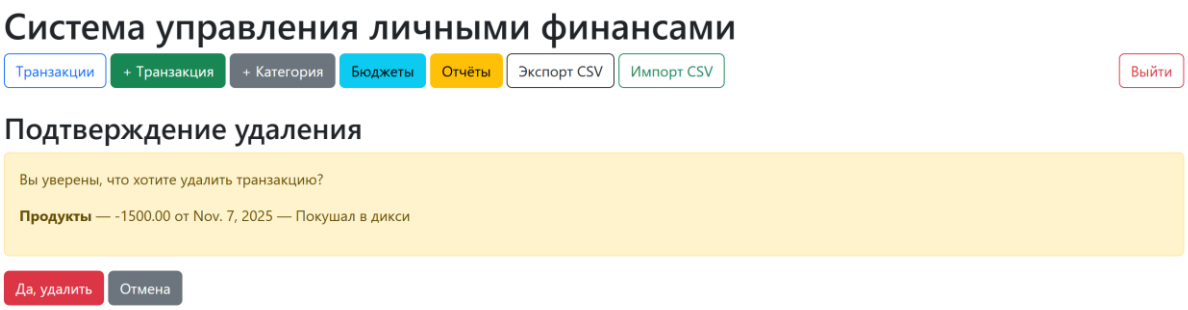

Рисунок 10 – Подтверждаем удаление транзакции

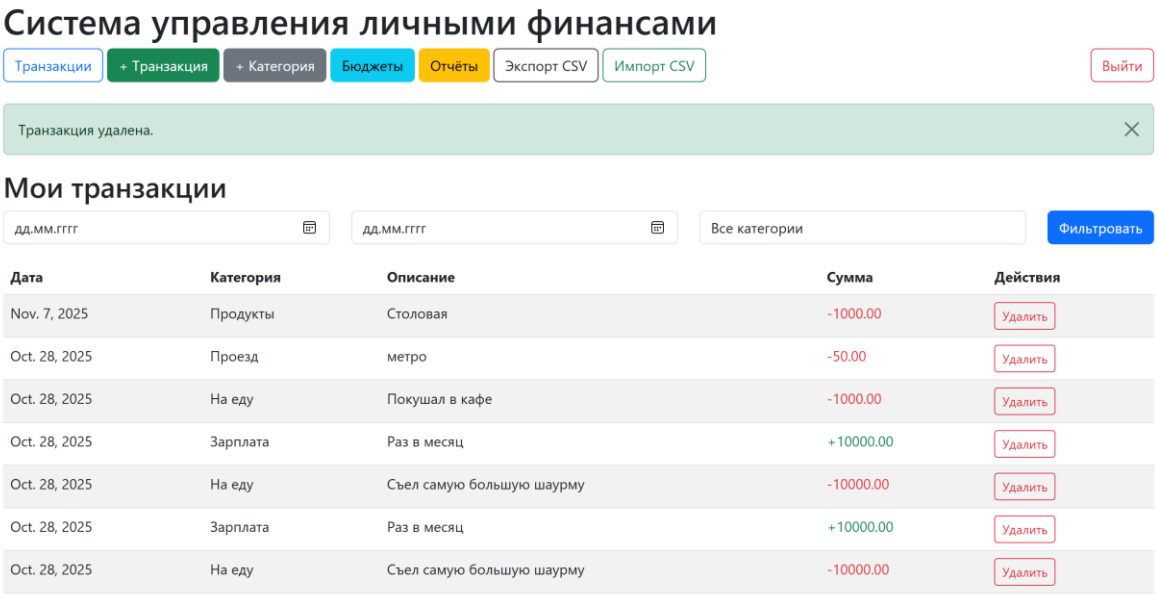

Рисунок 11 – Транзакция удалена

Тест-кейс 4: Фильтрация

1. Добавить несколько транзакций за разные даты и категории.

2. Использовать фильтр по дате и категории — проверить, что отображаются только соответствующие записи.

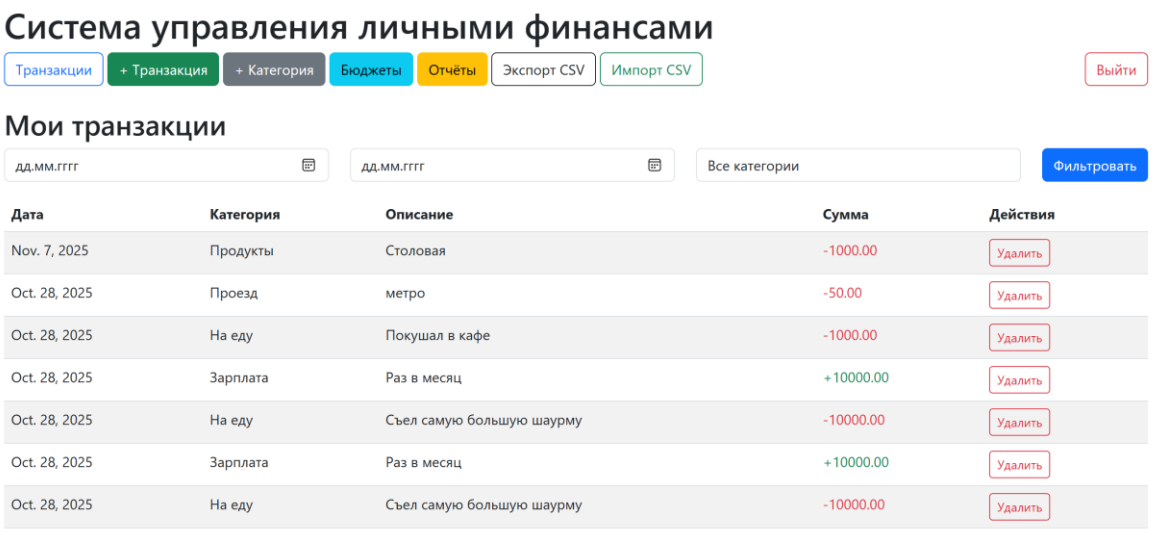

Рисунок 12 – Список транзакций

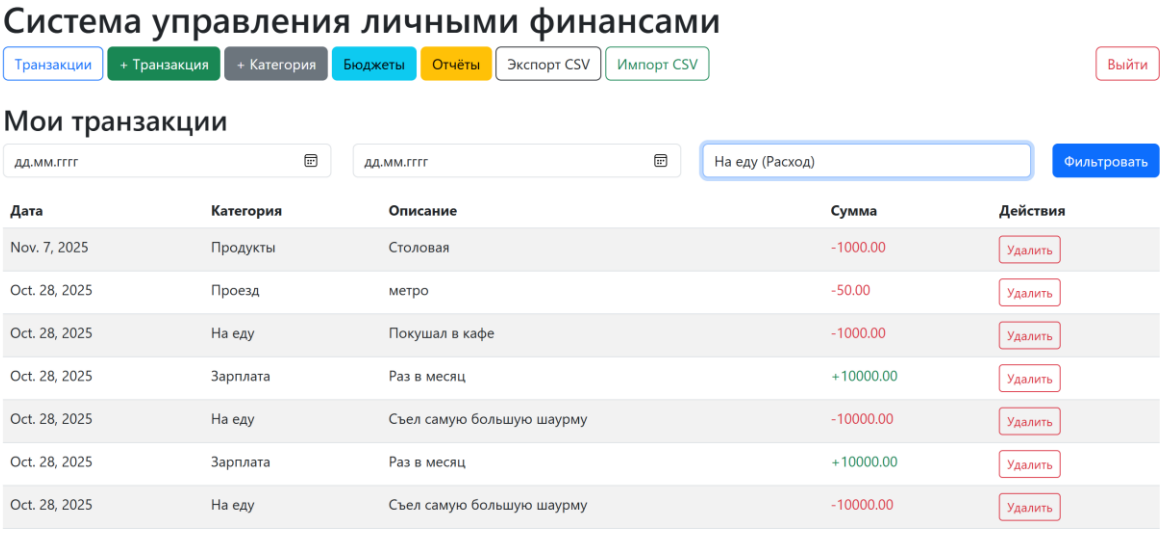

Рисунок 13 – Выбираем категорию фильтрации

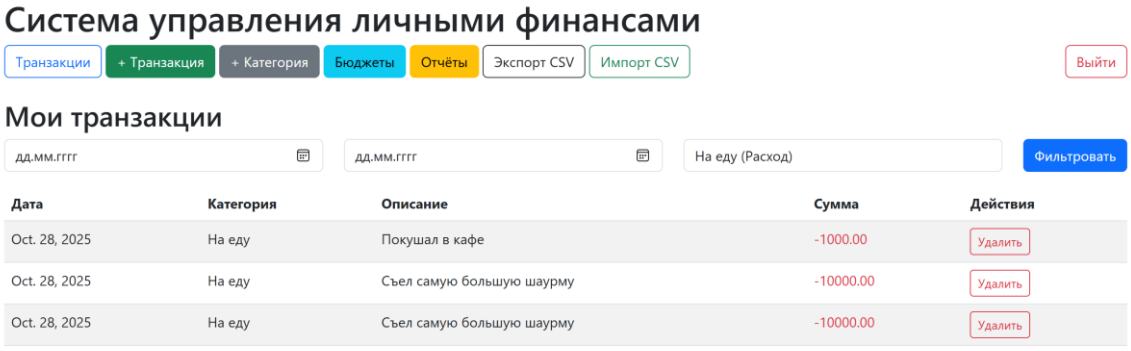

Рисунок 14 – Фильтрация работает успешно

Тест-кейс 5: Импорт и экспорт данных

1. Нажать «Экспорт CSV» — проверить, что скачивается файл transactions.csv.

2. Открыть файл — убедиться, что структура соответствует ожидаемой.

3. Перейти в «Импорт CSV», загрузить корректный CSV-файл.

4. Убедиться, что транзакции из файла появились в списке.

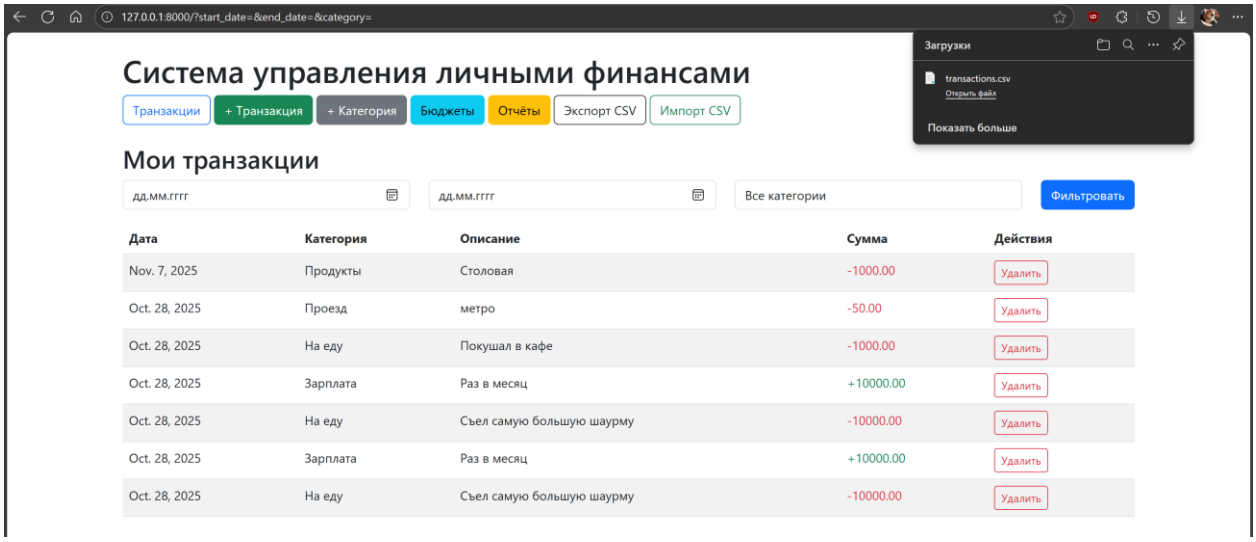

Рисунок 15 – Экспорт транзакций

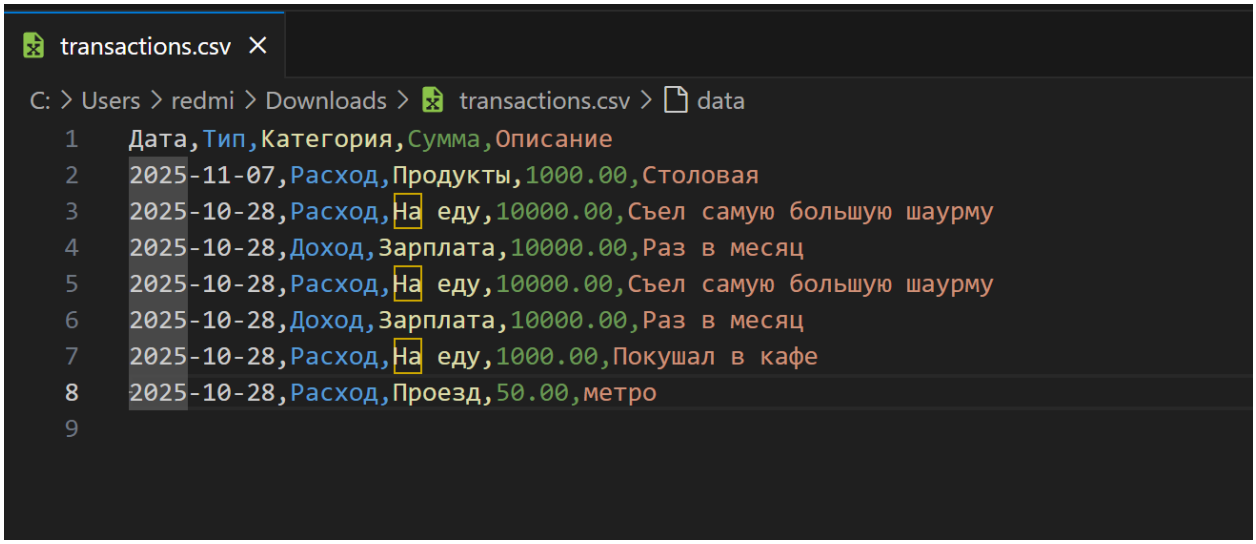

Рисунок 16 – Структура transactions.csv

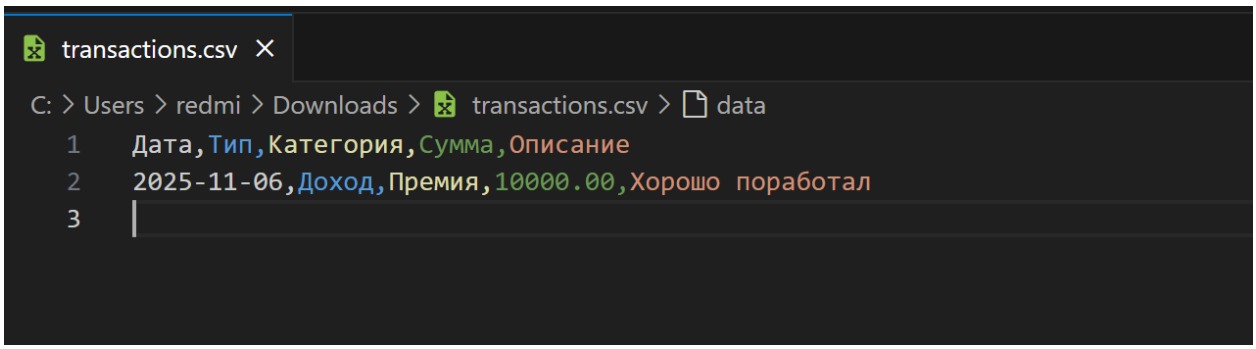

Рисунок 17 – Импортируемый файл

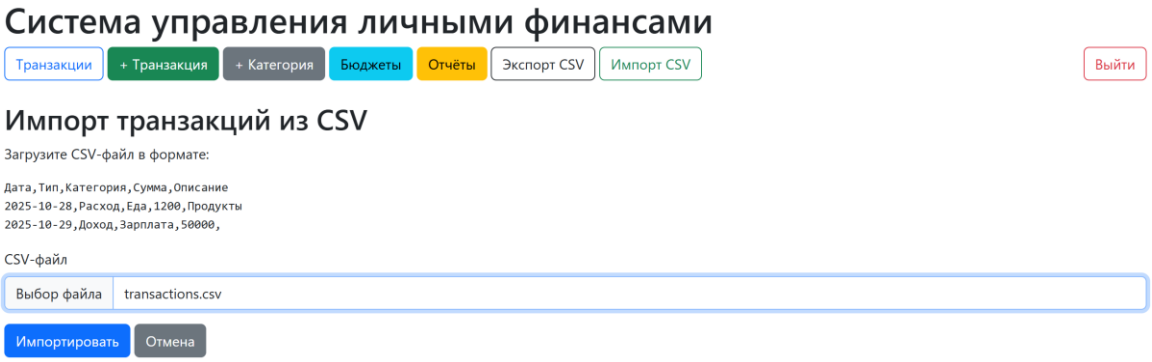

Рисунок 18 – Импорт transactions.csv

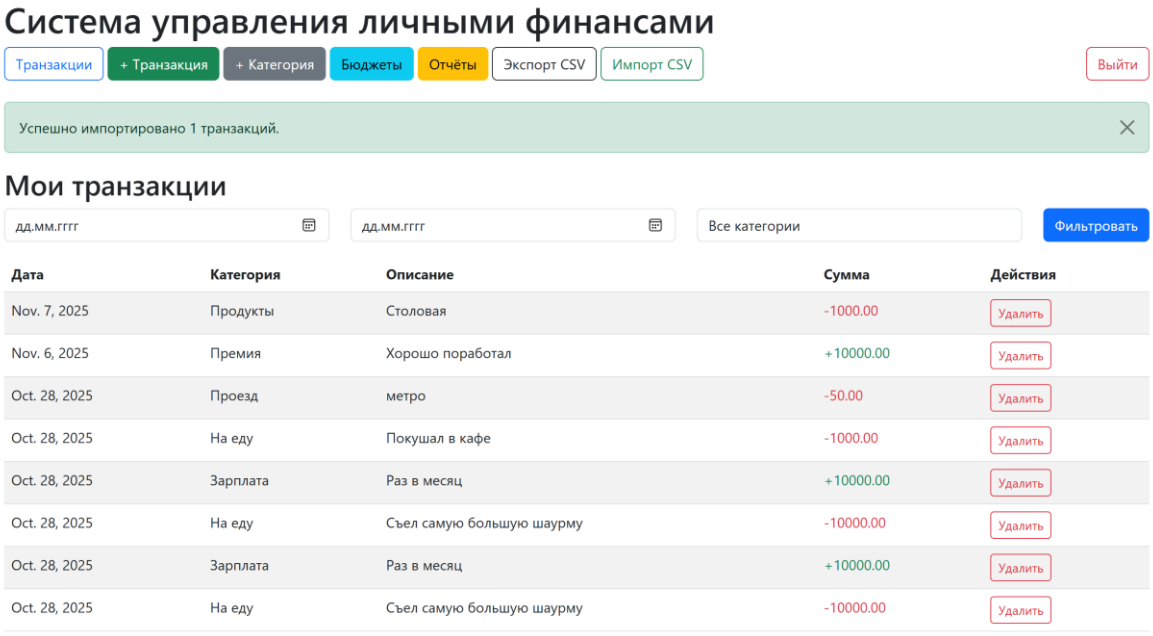

Рисунок 19 – Импорт успешно завершён

5.4. Критерии успешной проверки

Программа считается прошедшей проверку, если: все автоматизированные тесты завершаются успешно, все ручные тест-кейсы выполняются без ошибок, интерфейс корректно отображается в поддерживаемых браузерах, нет критических сбоев (ошибок 500) при стандартном использовании, Данные одного пользователя недоступны другому (проверяется через создание второго аккаунта).

Таким образом, предложенная методика обеспечивает всестороннюю проверку функциональности, надёжности и безопасности программы.

6. Руководство пользователя

6.1. Назначение документа

Настоящее руководство предназначено для конечных пользователей программы «Система управления личными финансами». Оно содержит пошаговые инструкции по установке, настройке и повседневному использованию приложения для учёта доходов и расходов.

6.2. Первый запуск и вход в систему

После установки программы (см. раздел 4) запустите веб-сервер командой:

python manage.py runserver

Откройте в браузере адрес:

http://127.0.0.1:8000/accounts/login/

Введите имя пользователя и пароль, созданные при выполнении команды createsuperuser.

После успешного входа вы будете перенаправлены на главную страницу — список транзакций.

6.3. Создание категорий операций

Категории позволяют классифицировать ваши финансовые операции.

Чтобы создать категорию:

1. Нажмите кнопку «+ Категория» в верхнем меню.

2. В открывшейся форме укажите:

3. Название: например, «Зарплата», «Еда», «Транспорт».

4. Тип: выберите «Доход» или «Расход».

5. Нажмите «Сохранить».

6. Созданные категории будут доступны при добавлении транзакций.

6.4. Добавление транзакций

Транзакция — это запись о доходе или расходе.

Чтобы добавить транзакцию:

1. Нажмите кнопку «+ Транзакция».

2. Заполните форму:

3. Сумма: положительное число (например, 1500).

4. Категория: выберите из списка созданных ранее.

5. Описание (опционально): например, «Покупка в магазине».

6. Дата: по умолчанию — сегодня, но можно выбрать любую дату.

7. Нажмите «Сохранить».

Транзакция появится в списке на главной странице. Доходы отображаются зелёным цветом со знаком «+», расходы — красным со знаком «–».

6.5. Просмотр и фильтрация транзакций

На главной странице отображаются все ваши транзакции в обратном хронологическом порядке.

Для фильтрации:

1. Укажите диапазон дат в полях «С даты» и «По дату».

2. При необходимости выберите конкретную категорию из выпадающего списка.

3. Нажмите «Фильтровать».

4. Чтобы сбросить фильтр, нажмите кнопку «Сбросить» или перейдите по ссылке на главную страницу.

6.6. Удаление транзакций

Если вы допустили ошибку при вводе, транзакцию можно удалить:

1. В списке транзакций найдите нужную запись.

2. Нажмите кнопку «Удалить» в последней колонке.

3. Подтвердите действие во всплывающем окне.

4. Внимание: удаление невозможно отменить.

6.7. Импорт и экспорт данных

Экспорт в CSV:

1. Нажмите кнопку «Экспорт CSV» в меню.

2. Браузер скачает файл transactions.csv.

Файл можно открыть в Excel, Google Таблицах или текстовом редакторе.

Формат файла:

Дата,Тип,Категория,Сумма,Описание

Импорт из CSV:

1. Перейдите в раздел «Импорт CSV».

2. Нажмите «Выберите файл» и укажите CSV-файл в требуемом формате.

3. Нажмите «Импортировать».

Важно: файл должен содержать заголовок и данные в указанном порядке. Категории, отсутствующие в системе, будут созданы автоматически.

6.8. Выход из системы

Для завершения работы:

1. Нажмите кнопку «Выйти» в правом верхнем углу.

2. Вы будете перенаправлены на страницу входа.

Вы можете войти под другим аккаунтом, если он создан в системе.

6.9. Техническая поддержка

В случае возникновения проблем:

- Убедитесь, что вы используете поддерживаемый браузер.
- Проверьте наличие интернета (для отображения стилей и диаграмм).
- При ошибках сохраните текст сообщения и обратитесь к разработчику.

Программа не требует постоянного подключения к интернету и может использоваться полностью автономно.

Заключение

В ходе выполнения курсовой работы была успешно разработана и документирована веб-система «Управление личными финансами», предназначенная для автоматизации учёта доходов и расходов, планирования бюджетов и анализа финансовой активности. Программа реализована с использованием современных подходов к разработке программного обеспечения: применена архитектура MVT на базе фреймворка Django, обеспечена модульность кода, выделен слой бизнес-логики, реализована поддержка многопользовательского режима с полной изоляцией данных.

Разработанное приложение отвечает поставленным целям и решает все заявленные задачи: пользователь может создавать собственные категории операций, добавлять и управлять транзакциями, экспортировать и импортировать данные в формате CSV, а также визуализировать финансовую информацию с помощью диаграмм. Особое внимание уделено безопасности — все операции привязаны к авторизованному пользователю, пароли хранятся в хэшированном виде, а формы защищены от CSRF-атак.

Программа протестирована как автоматизированными unit-тестами, так и ручными сценариями использования. Все функциональные требования подтверждены успешным прохождением тестовых кейсов, а архитектура поддерживает дальнейшее расширение функционала — включая добавление поддержки валют, уведомлений, мобильного API и прогнозной аналитики.

Таким образом, проект не только соответствует требованиям учебной дисциплины, но и представляет собой полноценное, готовое к использованию решение для повышения финансовой грамотности и эффективного управления личным бюджетом в автономном режиме без зависимости от сторонних облачных сервисов.

Список использованной литературы

1. Django Software Foundation. Django documentation [Электронный ресурс]. — Режим доступа: https://docs.djangoproject.com/en/5.0/ , свободный. — Дата обращения: 07.11.2025.
2. Mozilla Developer Network. Web technology reference documentation: HTML, CSS, JavaScript [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/ , свободный. — Дата обращения: 07.11.2025.
3. Chart.js: Open-source HTML5 Charts for Your Web Application [Электронный ресурс]. — Режим доступа: https://www.chartjs.org/ , свободный. — Дата обращения: 07.11.2025.
4. Python Software Foundation. Python 3.12 documentation [Электронный ресурс]. — Режим доступа: https://docs.python.org/3/ , свободный. — Дата обращения: 07.11.2025.
5. Дицель Н. С. Система управления личными финансами [Электронный ресурс] / Н. С.Дицель. — Режим доступа: https://github.com/Relax1205/kursach_5 , свободный. — Дата обращения: 07.11.2025.
6. ГОСТ 19.101–77. Единая система программной документации (ЕСПД). Схемы алгоритмов, программ, данных и систем. — Введ. 1978-01-01. — М.: Издательство стандартов, 1977. — 10 с.
7. ГОСТ 34.601–90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. — Введ. 1992-01-01. — М.: Издательство стандартов, 1990. — 15 с.