Введение

HomeBudget — десктопное приложение для операционной системы Windows, представляющий собой систему учета личных финансов.

Актуальность разработки обусловлена современными тенденциями повышения финансовой грамотности населения и растущей потребностью в инструментах для контроля расходов. В условиях экономической нестабильности и увеличения количества финансовых операций, совершаемых ежедневно, приложение становится незаменимым помощником для тех, кто стремится к осознанному распоряжению своими денежными средствами. Особую значимость проект приобретает благодаря своей простоте использования и доступности для широкого круга пользователей.

Основная цель проекта заключается в создании удобного и функционального инструмента для систематического учета личных финансов. Приложение должно предоставлять пользователям возможность не только фиксировать доходы и расходы, но и анализировать свою финансовую деятельность через систему отчетов и визуализации данных, что в конечном итоге способствует более грамотному распределению бюджета.

Целевая аудитория приложения включает в себя различные группы пользователей: от студентов и молодых специалистов, только начинающих вести учет своих расходов, до семейных пар и фрилансеров, нуждающихся в постоянном контроле денежных потоков. Приложение будет особенно полезно тем, кто стремится к финансовой дисциплине и планированию долгосрочных сбережений.

Ключевые функции приложения охватывают весь цикл работы с личными финансами: от базовых операций регистрации и авторизации пользователей до сложных функций анализа данных. Основной функционал включает создание и редактирование финансовых записей, категоризацию расходов и доходов, формирование детализированных отчетов за различные периоды времени, а также возможность настройки интерфейса под индивидуальные предпочтения пользователя. Особое внимание уделено системе экспорта отчетов в популярные форматы для дальнейшей обработки.

Структура отчета отражает последовательность разработки приложения и включает несколько основных разделов. Вначале рассматриваются общие требования к проекту и его технические характеристики, затем подробно описывается процесс проектирования базы данных и пользовательского интерфейса. Особый раздел посвящен реализации функциональных возможностей приложения и методам тестирования его работоспособности. Завершается отчет анализом полученных результатов и выводами о степени соответствия готового продукта первоначальным требованиям технического задания.

1 Проектирование программного продукта

Проектирование программного продукта — это этап разработки, на котором создается архитектура, функциональность и пользовательский интерфейс будущего приложения. Оно определяет, как система будет работать, прежде чем начнется ее непосредственная реализация в коде.

1.1 Диаграмма связей

Представленная диаграмма связей (рисунок 1) отражает архитектуру приложения для учета личных финансов, демонстрируя взаимосвязи между его основными функциональными модулями. В центре системы находится главное меню, выступающее в качестве ключевого узла навигации. Из него пользователь получает доступ к пяти основным разделам: модулю внесения и редактирования доходов, разделу учета расходов, компоненту управления категориями операций, блоку формирования аналитических отчетов и разделу персонализации настроек приложения.

Вход в систему начинается с меню входа, где пользователю предлагается два варианта действий — авторизация существующих пользователей или регистрация нового аккаунта. После успешного прохождения процедуры аутентификации открывается доступ к главному меню, которое обеспечивает интуитивно понятную навигацию между всеми функциональными разделами. Каждый из основных модулей связан с главным меню двусторонней связью, что позволяет не только быстро переходить к нужному разделу, но и возвращаться обратно после выполнения необходимых операций.

Архитектура приложения построена таким образом, чтобы обеспечить логичную последовательность работы с финансовыми данными: от внесения информации о доходах и расходах через соответствующие модули до анализа накопленных данных с помощью системы отчетов. Особое внимание уделено унификации интерфейса - все функциональные блоки спроектированы в едином стиле и связаны через центральное меню, что значительно упрощает процесс освоения приложения новыми пользователями. Отдельный модуль настроек позволяет адаптировать интерфейс под индивидуальные предпочтения, повышая удобство ежедневного использования системы.

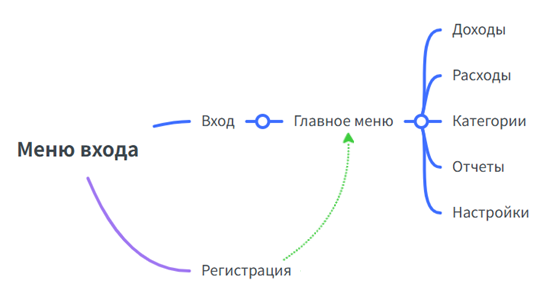


Рисунок 1 — Диаграмма связей приложения

1.2 Диаграмма сценариев

На представленной диаграмме сценариев (рисунок 2) представлено взаимодействие двух типов пользователей с системой — гостя и зарегистрированного пользователя. Гость может выполнять только базовые операции: регистрацию новой учетной записи с указанием необходимых персональных данных и последующую авторизацию в системе. После успешного входа в систему статус гостя изменяется на авторизованного пользователя, что открывает доступ к полному функционалу приложения.

Для авторизованных пользователей предусмотрены расширенные возможности работы с системой. Основной цикл взаимодействия включает ввод финансовых данных о доходах и расходах, настройку параметров учета, генерацию различных отчетов и просмотр накопленной информации в удобном формате визуализации. Все функциональные возможности взаимосвязаны и образуют логичную последовательность действий: от первичного ввода данных через их систематизацию к анализу и визуализации результатов.

Особенностью представленной схемы является четкое разделение прав доступа между гостями и зарегистрированными пользователями, а также последовательное расширение функциональных возможностей после прохождения процедуры авторизации. Архитектура взаимодействия построена по принципу постепенного усложнения — от простых операций регистрации к комплексной работе с финансовыми данными, что обеспечивает комфортное освоение системы новыми пользователями. Все элементы диаграммы связаны между собой, образуя целостную картину пользовательских сценариев работы с приложением.

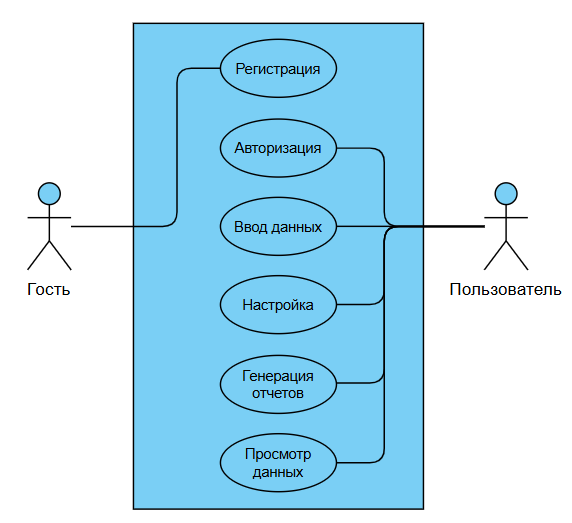


Рисунок 2 — Диаграмма сценариев приложения

1.3 Прототипирование и дизайн

Прототипирование помогает быстро проверить работоспособность идеи через создание упрощённых моделей, а дизайн превращает эти решения в удобные, функциональные и визуально привлекательные продукты.

На рисунке 3 представлен прототип, который демонстрирует логику работы приложения. Из начального экрана есть переход либо на страницу входа, либо на страницу регистрации. После входа/регистрации пользователь попадает на главную страницу. С этой страницы сразу обеспечивается доступ ко всем остальным страницам приложения. На этой же вкладке расположена кнопка выхода из приложения.

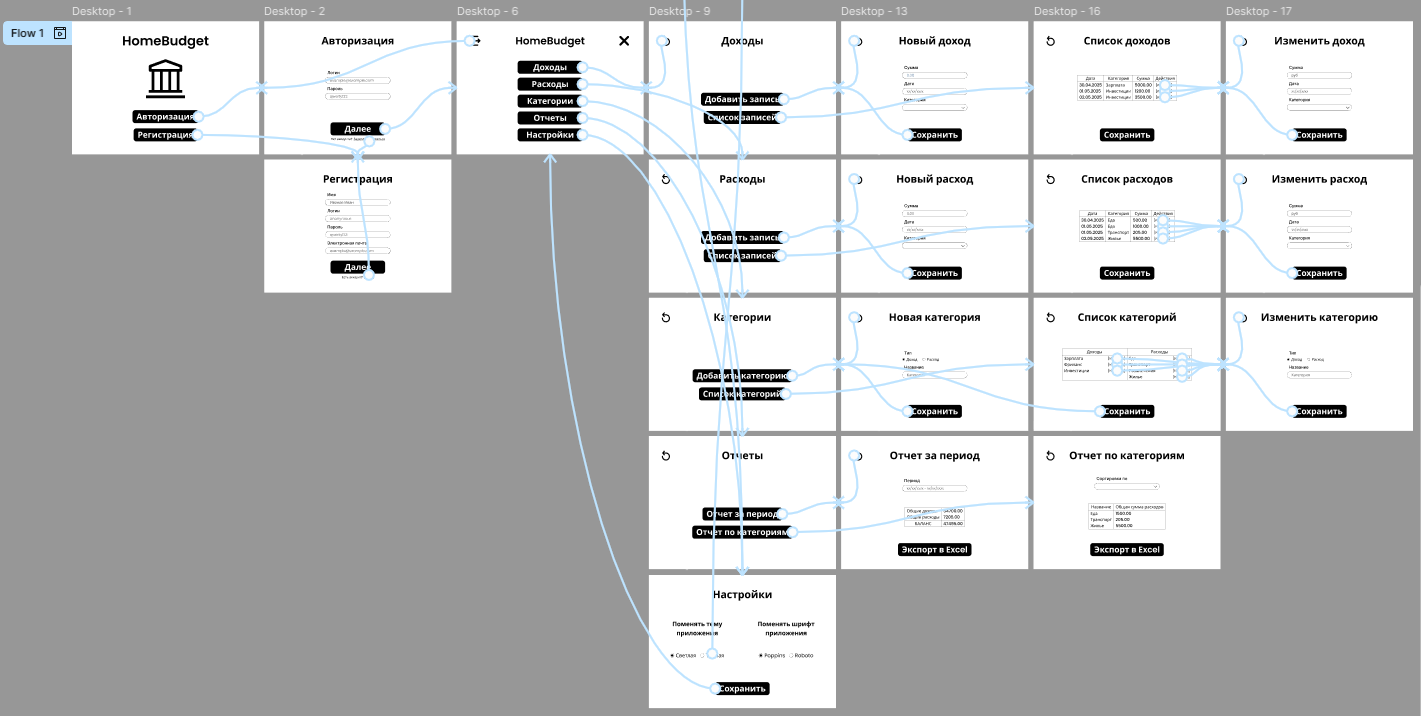


Рисунок 3 — прототип приложения

На рисунке 4 представлен макет начального экрана приложения с описанием приложения. В интерфейсе представлена кнопка, которая предусматривает переход на страницу входа.



Рисунок 4 — макет начального экрана

На рисунке 5 представлен макет главной страницы. В интерфейсе представлено меню перехода между страницами, кнопка выхода из аккаунта и кнопка выхода из приложения.



Рисунок 5 — макет главной страницы

На рисунке 6 представлен макет страницы создания отчётов за выбранный период темной темы. В интерфейсе представлены кнопка возврата на предыдущую страницу, текстовое поле для временного периода, таблица с данными за соответствующий период и кнопка экспорта отчета.

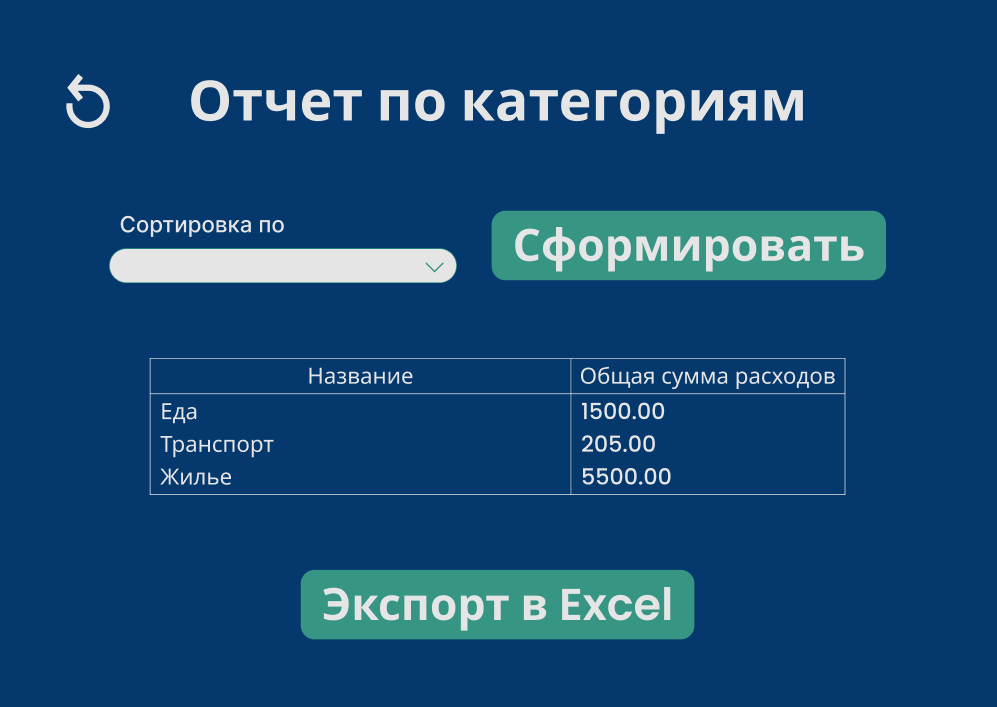


Рисунок 6 — макет страницы создания отчетов

На рисунке 7 представлен QR-код, ведущий на страницу с доступом к интерактивной версии прототипа приложения.

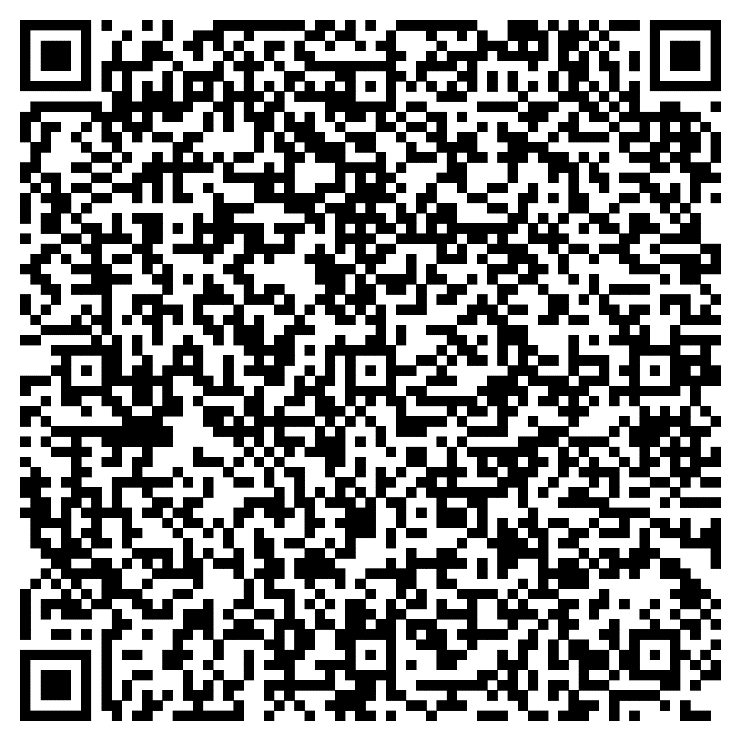


Рисунок 7 — QR-код к прототипу

1.4 Проектирование БД

Проектирование БД – это процесс создания структуры базы данных, определяющий таблицы, поля, типы данных и связи между ними. Цель - обеспечить эффективное хранение, доступ и управление данными, а также соответствие требованиям приложения.

На рисунке 8 представлена ER-диаграмма базы данных HomeBudget, изображены четыре таблицы: "Users", "Categories", "Income" и "Expense". Связи между таблицами показывают их взаимоотношения: "Users" и "Income" связаны отношением "один ко многим" (один пользователь может иметь много записей доходов). "Users" и "Expense" связаны отношением "один ко многим" (один пользователь может иметь много записей расходов). "Categories" и "Income" связаны отношением "один ко многим” (одна категория может относиться ко многим доходам). "Categories" и "Expense" связаны отношением "один ко многим” (одна категория может относиться ко многим расходам).

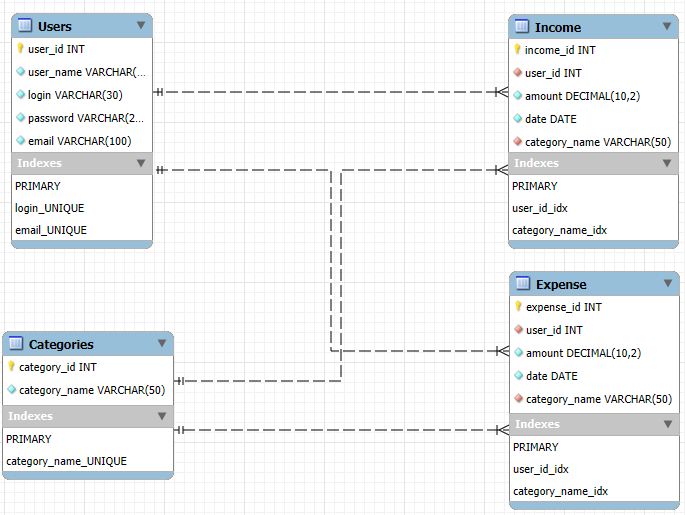


Рисунок 8 – Диаграмма “сущность-связь”

2.1 Описание средств разработки

Java с платформой JavaFX выбрана в качестве основного инструмента для создания графического интерфейса. JavaFX предоставляет богатый набор компонентов такие как кнопки, текстовые поля, списки и фреймы, что позволяет разрабатывать современные оконные приложения с интерактивными элементами. Платформа интегрируется с Java, обеспечивая удобную связь между пользовательским интерфейсом и бизнес-логикой приложения.

Visual Paradigm применялся для проектирования структуры базы данных. С его помощью были созданы ER-диаграммы, отображающие связи между таблицами. Инструмент поддерживает генерацию SQL-скриптов на основе схемы, что ускоряет процесс настройки базы данных.

Figma использовалась для разработки макетов интерфейса. В этом инструменте создавались прототипы экранов с детализацией расположения элементов, цветовой схемы и типографики. Готовые макеты служили основой для реализации интерфейса в JavaFX.

MindMeister применялся для структурирования идей и планирования функциональности приложения. С помощью данного инструмента удалось визуализировать комплексную архитектуру системы, отображая не только основные функциональные модули, но и сложные взаимосвязи между ними. Ментальные карты позволили зафиксировать ключевые сущности предметной области с их атрибутами и характеристиками, что стало основой для последующего проектирования базы данных.

MySQL применялся в качестве реляционной системы управления базами данных для хранения и обработки структурированной информации. В проекте реализована база данных с нормализованной структурой, включающей таблицы Users, Categories, Income и Expense.

MySQL Workbench использовался как основной инструмент для администрирования базы данных, выполнения SQL-запросов и визуального проектирования структуры таблиц.

В данном проекте технологический стек включает Java с JavaFX для реализации кроссплатформенного графического интерфейса, Figma для проектирования UI/UX-макетов перед их программной интеграцией, а также MySQL и MySQL Workbench, обеспечивающие надежное хранение данных, проектирование реляционной схемы базы данных и выполнение SQL-запросов. Такая комбинация инструментов охватывает полный цикл разработки — от проектирования интерфейса и бизнес-логики до работы с данными, формируя согласованную систему с возможностью дальнейшего масштабирования функциональности.

2.2 Создание интерфейса пользователя

На рисунке 9 представлен стартовый экран приложения, выполняющий ключевую функцию аутентификации пользователей. Этот экран предлагает два основных пути взаимодействия: авторизацию для существующих пользователей и процесс регистрации для новых.

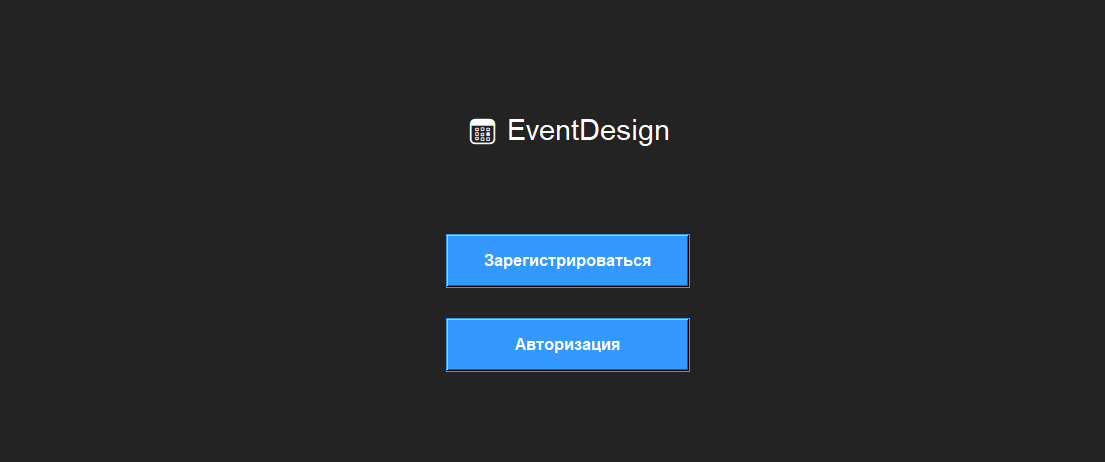


Рисунок 9 — Стартовый экран приложения

На рисунке 10 представлен экран категорий мероприятий, позволяющий добавить новое мероприятие с его описанием, возможностью редактирования и удаления. Также имеется кнопка “Назад”, которая возвращает на главный экран приложения.

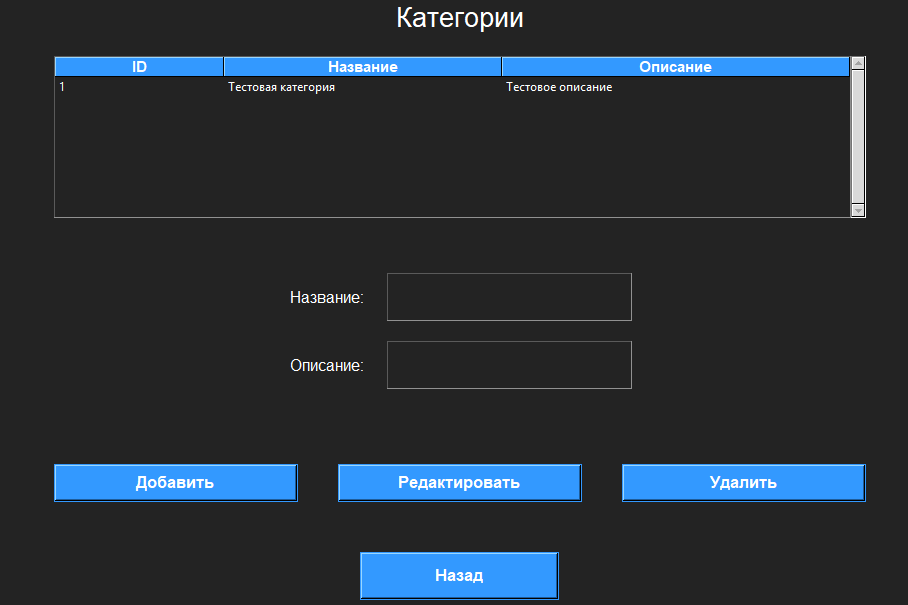


Рисунок 10 — Вкладка “Категории”

На рисунке 11 изображен экран с избранными мероприятиями. В верхней части интерфейса расположена таблица со следующими ячейками “Название”, “Категория”, ”Место”, ”Дата”, ”Описание” и заголовок "Избранное", обозначающий раздел для персональных подборок пользователя.

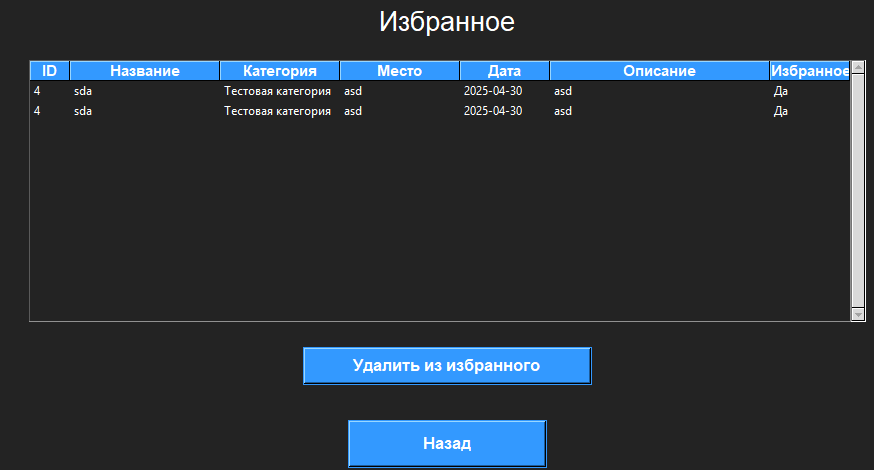


Рисунок 11 — Вкладка “Избранное”

2.3 Взаимодействие с базой данных

На рисунке 12 представлена структура проекта "EventDesign", содержащая ключевые элементы для работы с базой данных и приложением. В корне проекта находятся файлы db\_config.py с параметрами подключения к MySQL, db\_utils.py с функциями для взаимодействия с БД, и main.py — основной скрипт приложения. Папка venv хранит виртуальное окружение Python, файл requirements.txt перечисляет зависимости проекта. Остальные элементы, такие как служебные папки и файлы, обеспечивают вспомогательную функциональность, не затрагивая напрямую логику работы с базой данных.

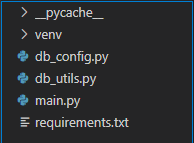


Рисунок 12 — Структура приложения

1. Тестирование программного продукта

3.1 Описание используемых видов тестирования

Функциональное тестирование — процесс обеспечения качества в рамках цикла разработки программного обеспечения, необходимый для проверки реализуемости функциональных требований, согласно спецификации тестируемого программного обеспечения.

Ручное тестирование – это процесс проверки программного обеспечения вручную, без использования автоматизированных скриптов. Тестировщик выполняет тест-кейсы, имитируя действия пользователя, чтобы выявить ошибки в интерфейсе, логике или функциональности.

3.2 Тест-план

На таблице 1 указан план тестирования программного продукта

Таблица 1 – План тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| Тест- кейс,  № | Название | Описание | Метод тестирования |
| 1. | Регистрация с валидными данными | Регистрация нового пользователя с введением корректных данных | Ручное |
| 2. | Регистрация с невалидными данными | Регистрация нового пользователя с введением некорректных данных | Ручное |
| 3. | Регистрация с пустыми полями | Регистрация нового пользователя без введения данных | Ручное |
| 4. | Авторизация существующего пользователя | Авторизация зарегистрированного пользователя с введением корректных данных | Ручное |
| 5. | Авторизация с невалидными данными | Авторизация зарегистрированного пользователя с введением  некорректных данных | Ручное |
| 6. | Авторизация с пустыми полями | Авторизация зарегистрированного пользователя с пустыми полями | Ручное |
| 7. | Ввод данных | Ввод данных пользователем в форму | Ручное |
| 8. | Генерация отчетов | Процесс создания и предоставления данных о мероприятиях | Ручное |

1. Тест-кейсы

На таблице 2 указан тест кейс “Регистрации с валидными данными”.

Таблица 2 – Тест-кейс №1 «Регистрации с валидными данными

|  |  |  |
| --- | --- | --- |
| Название | Регистрация с валидными данными | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | * Пройден * Провален * Заблокирован |
| Предусловие | Аккаунт не зарегистрирован | |
| Шаги теста |  | |
| Запустить программу | Успешный запуск программы | Пройден |
| Открыть окно регистрации | Открыта форма регистрации | Пройден |
| Ввести корректные данные во все поля | Данные вводятся и корректно отображаются | Пройден |
| Нажать на кнопку  «Регистрация» | Отображается оповещение о регистрации пользователя с введенным именем, произошел переход в главное меню. | Пройден |

На таблице 3 указан тест кейс “Регистрации с невалидными данными”.

Таблица 3 – Тест-кейс №2 «Регистрации с невалидными данными».

|  |  |  |
| --- | --- | --- |
| Название | Регистрация с невалидными данными | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | * Пройден * Провален   -Заблокирован |
| Предусловие | Аккаунт не зарегистрирован | |
| Шаги теста |  | |
| Запустить программу | Успешный запуск программы | Пройден |
| Открыть окно регистрации | Открыта форма регистрации | Пройден |
| Ввести некорректные данные во все поля | Данные вводятся и отображаются | Пройден |
| Нажать на кнопку «Регистрация» | Отображается оповещение о неверных данных, переход на страницу входа | Пройден |

На таблице 4 указан тест кейс “Регистрация с пустыми полями”

Таблица 4 – Тест-кейс №3 «Регистрация с пустыми полями».

|  |  |  |
| --- | --- | --- |
| Название | Регистрация с пустыми полями | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | * Пройден * Провален * ​   Заблокирован |
| Предусловие | Аккаунт не зарегистрирован | |
| Шаги теста |  | |
| Запустить программу | Успешный запуск программы | Пройден |
| Открыть окно регистрации | Открыта форма регистрации | Пройден |
| Ввести корректные данные во все поля | Данные не вводятся | Пройден |
| Нажать на кнопку  «Регистрация» | Отображается оповещение о том что важно заполнить поля, переход на окно входа не произошел. | Провален |

На таблице 5 указан тест кейс “Авторизация существующего пользователя”

Таблица 5 – Тест-кейс №4 «Авторизация существующего пользователя».

|  |  |  |
| --- | --- | --- |
| Название | Авторизация существующего пользователя | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | * Пройден * Провален * Заблокирован |
| Предусловие | Аккаунт зарегистрирован | |
| Шаги теста |  | |
| Запустить программу | Успешный запуск программы | Пройден |
| Открыть окно входа | Открыта форма входа в аккаунт | Пройден |
| Ввести корректные данные во все поля | Данные вводятся и корректно отображаются | Пройден |
| Нажать на кнопку  «Войти» | Отображается оповещение о том что вход в аккаунт успешно выполнен | Пройден |

На таблице 6 указан тест кейс “Авторизация с невалидными данными”

Таблица 6 – Тест-кейс №5 «Авторизация с невалидными данными».

|  |  |  |
| --- | --- | --- |
| Название | Авторизация существующего пользователя | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | * Пройден * Провален * Заблокирован |
| Предусловие | Аккаунт зарегистрирован | |
| Шаги теста |  | |
| Запустить программу | Успешный запуск программы | Пройден |
| Открыть окно входа | Открыта форма входа в аккаунт | Пройден |
| Ввести корректные данные во все поля | Данные вводятся и корректно отображаются | Пройден |
| Нажать на кнопку  «Войти» | Отображается оповещение о том что введены неверные логин и пароль | Пройден |

Таблица 7 – баг-репорт по тест-кейсу №3 «Регистрация с пустыми полями»

|  |  |
| --- | --- |
| Баг-репорт по тест-кейсу №3 «Регистрация с пустыми полями» | |
| Название ошибки | Некорректное сообщение при вводе пустых полей |
| Описание проблемы | При регистрации с пустыми полями приложение выдает ошибку |
| Шаги воспроизведения | |
|  | Перейти на страницу регистрации |
|  | Не заполнять поля |
|  | Нажать на кнопку «Регистрация» |
| Ожидаемый результат | Вывод сообщения «Поля должны быть заполнены» |
| Фактический результат | Сообщение не выводится |
| Серьезность | Незначительный |
| Приоритет | Высокий |
| Исправление ошибки | |
|  | Проверка корректности отображения сообщения |
|  | Проверка логики обработки если не вводятся данные |
|  | Исправление логики обработки если не вводятся данные |
| Повторное тестирование | |
|  | Повторить шаги тест-кейса с теми же данными |
|  | Убедиться, что оповещение об ошибке выводится пользователю |
|  | Убедиться, что переход на окно входа не происходит |
| Результат | Отображается сообщение «Поля должны быть» |

На таблице 8 указан тест кейс “Регистрации с невалидными данными”.

Таблица 8 – Повторный тест-кейс №3 «Регистрация с пустыми полями».

|  |  |  |
| --- | --- | --- |
| Название | Регистрация с пустыми полями | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | * Пройден * Провален * ​   Заблокирован |
| Предусловие | Аккаунт не зарегистрирован | |
| Шаги теста |  | |
| Запустить программу | Успешный запуск программы | Пройден |
| Открыть окно регистрации | Открыта форма регистрации | Пройден |
| Ввести корректные данные во все поля | Данные не вводятся | Пройден |
| Нажать на кнопку  «Регистрация» | Отображается оповещение о том что важно заполнить поля, переход на окно входа не произошел. | Пройден |

Заключение

Приложение "EventDesign" было разработано с целью создания удобного инструмента для управления и организации мероприятий, включая функционал регистрации и авторизации пользователей, ввода и редактирования данных, генерации отчетов, управления категориями и создания списка избранных событий. В целом, приложение успешно справляется с основными задачами, такими как регистрация и авторизация, ввод и редактирование данных, управление категориями и отображение избранных мероприятий, которые работают стабильно и организованы удобно. Однако необходимо улучшить пользовательский интерфейс для повышения удобства использования. С учетом объема выполненной работы, затраты времени и ресурсов можно считать оправданными, но оптимизация некоторых процессов разработки могла бы повысить эффективность. Для улучшения приложения можно было бы внедрить более современные фреймворки для разработки UI, оптимизировать работу с базами данных и добавить функции календарного отображения мероприятий, уведомлений и экспорта отчетов. В процессе работы были закреплены навыки использования Python и Tkinter для создания графического интерфейса, а также получен опыт работы с базами данных MySQL. Полученный опыт будет полезен в будущих проектах, связанных с разработкой десктопных приложений и работой с базами данных.

Приложение А

На листинге 1 представлен код, который создает окно настроек с заголовком "Настройки". Окно имеет темный фон с белым текстом, и соответствующими настройками темы приложения. В верхней части окна отображается заголовок "Настройки" с шрифтом Arial размером 20 пикселей. Ниже расположен блок выбора темы, содержащий текстовую метку "Тема:" и два переключателя (для выбора между темной и светлой темами).

Листинг 1 – Код создания страницы настроек

class SettingsFrame(ThemedFrame):

def \_\_init\_\_(self, master, on\_back):

super().\_\_init\_\_(master, bg=master['bg'])

# кнопка назад

back\_btn = ttk.Button(

self,

text="Назад",

command=on\_back,

style='Rounded.TButton',

width=20

)

# размещаю кнопку внизу

back\_btn.pack(side='bottom', pady=20, ipady=5)

tk.Label(self, text='Настройки', font=('Arial', 20), fg=master.style.lookup('TButton', 'foreground'), bg=master['bg']).pack(pady=10)

theme\_frame = tk.Frame(self, bg=master['bg'])

theme\_frame.pack(pady=10)

tk.Label(theme\_frame, text='Тема:', fg=master.style.lookup('TButton', 'foreground'), bg=master['bg']).pack(side='left')

self.theme\_var = tk.StringVar(value=master.theme)

ttk.Radiobutton(theme\_frame, text='Тёмная', variable=self.theme\_var, value='dark', command=lambda: master.set\_theme('dark')).pack(side='left', padx=5)

ttk.Radiobutton(theme\_frame, text='Светлая', variable=self.theme\_var, value='light', command=lambda: master.set\_theme('light')).pack(side='left', padx=5)

def update\_theme(self, bg, fg, btn\_bg, btn\_fg):

super().update\_theme(bg, fg, btn\_bg, btn\_fg)

Приложение Б

Приложение Б

На листинге 1 представлена база данных для системы учета личных финансов. Этот код создает четыре таблицы для управления пользователями, категориями, доходами и расходами.

Листинг 1 — Код таблицы MySQL

CREATE TABLE IF NOT EXISTS `HomeBudget`.`Users` (

`user\_id` INT NOT NULL AUTO\_INCREMENT,

`user\_name` VARCHAR(50) NOT NULL,

`login` VARCHAR(30) NOT NULL,

`password` VARCHAR(255) NOT NULL,

`email` VARCHAR(100) NOT NULL,

PRIMARY KEY (`user\_id`),

UNIQUE INDEX `login\_UNIQUE` (`login` ASC) VISIBLE,

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE)

CREATE TABLE IF NOT EXISTS `HomeBudget`.`Categories` (

`category\_id` INT NOT NULL AUTO\_INCREMENT,

`category\_name` VARCHAR(50) NOT NULL,

PRIMARY KEY (`category\_id`),

UNIQUE INDEX `category\_name\_UNIQUE` (`category\_name` ASC) VISIBLE)

CREATE TABLE IF NOT EXISTS `HomeBudget`.`Income` (

`income\_id` INT NOT NULL AUTO\_INCREMENT,

`user\_id` INT NOT NULL,

`amount` DECIMAL(10,2) NOT NULL,

`date` DATE NOT NULL,

`category\_name` VARCHAR(50) NOT NULL,

PRIMARY KEY (`income\_id`),

INDEX `user\_id\_idx` (`user\_id` ASC) VISIBLE,

INDEX `category\_name\_idx` (`category\_name` ASC) VISIBLE,

CONSTRAINT `user\_id`

FOREIGN KEY (`user\_id`)

REFERENCES `HomeBudget`.`Users` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `category\_name`

FOREIGN KEY (`category\_name`)

REFERENCES `HomeBudget`.`Categories` (`category\_name`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

CREATE TABLE IF NOT EXISTS `HomeBudget`.`Expense` (

`expense\_id` INT NOT NULL AUTO\_INCREMENT,

`user\_id` INT NOT NULL,

`amount` DECIMAL(10,2) NOT NULL,

`date` DATE NOT NULL,

`category\_name` VARCHAR(50) NOT NULL,

PRIMARY KEY (`expense\_id`),

INDEX `user\_id\_idx` (`user\_id` ASC) VISIBLE,

INDEX `category\_name\_idx` (`category\_name` ASC) VISIBLE,

CONSTRAINT `user\_id`

FOREIGN KEY (`user\_id`)

REFERENCES `HomeBudget`.`Users` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `category\_name`

FOREIGN KEY (`category\_name`)

REFERENCES `HomeBudget`.`Categories` (`category\_name`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

Приложение В

## 1. Описание приложения

EventDesign – это десктопное приложение для Windows, предназначенное для планирования и управления мероприятиями. Приложение позволяет пользователям создавать, редактировать и отслеживать различные события, организовывать их по категориям и формировать отчеты.

## 2. Цель использования

* Планирование и организация мероприятий
* Управление категориями мероприятий
* Создание и ведение списка избранных мероприятий
* Генерация отчетов в различных форматах
* Персонализация интерфейса (светлая/темная тема)

## 3. Шаги по скачиванию и установке

1. Скачайте архив с проектом
2. Установите Python 3.x с официального сайта
3. Установите MySQL Server
4. Создайте виртуальное окружение:

## 4. Пошаговая инструкция для регистрации

1. При запуске приложения нажмите кнопку "Зарегистрироваться"
2. Заполните обязательные поля:

* Имя (только буквы и пробелы)
* Логин (буквы, цифры и символ подчеркивания)
* Пароль (минимум 6 символов, должен содержать цифру)
* Email (должен содержать символ @)

1. Нажмите кнопку "Зарегистрироваться"

## 5. Инструкция по входу в систему

1. На главном экране нажмите "Авторизация"
2. Введите логин и пароль
3. Нажмите кнопку "Вход"

## 6. Ключевые функции

### Управление мероприятиями

* Создание нового мероприятия:
* Нажмите "Мероприятия" в главном меню
* Заполните форму (название, категория, дата, место)
* Нажмите "Добавить"
* Редактирование мероприятия:
* Выберите мероприятие в списке
* Внесите изменения в форму
* Нажмите "Сохранить"
* Удаление мероприятия:
* Выберите мероприятие
* Нажмите "Удалить"

### Управление категориями

* Создание категории:
* Нажмите "Категории" в главном меню
* Введите название и описание
* Нажмите "Добавить"
* Редактирование категории:
* Выберите категорию
* Внесите изменения
* Нажмите "Сохранить"

### Работа с избранным

* Добавление в избранное:
* Выберите мероприятие
* Отметьте галочку "Избранное"
* Просмотр избранного:
* Нажмите "Избранное" в главном меню

### Генерация отчетов

* Выберите "Отчеты" в главном меню
* Выберите тип отчета (Excel или PDF)
* Укажите период и категорию
* Нажмите "Сформировать отчет"

### Настройки

* Смена темы:
* Нажмите "Настройки" в главном меню
* Выберите "Светлая" или "Темная" тема

## 7. Контактная информация

Для получения поддержки обращайтесь:

* ФИО: Ковалкин А.К.
* Группа: ИС-35
* Год: 2025