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. 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.

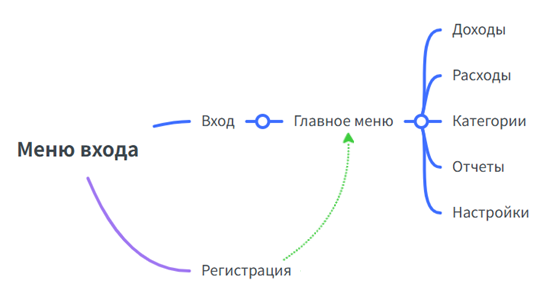


Рисунок 1 — Диаграмма связей приложения

1. 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.

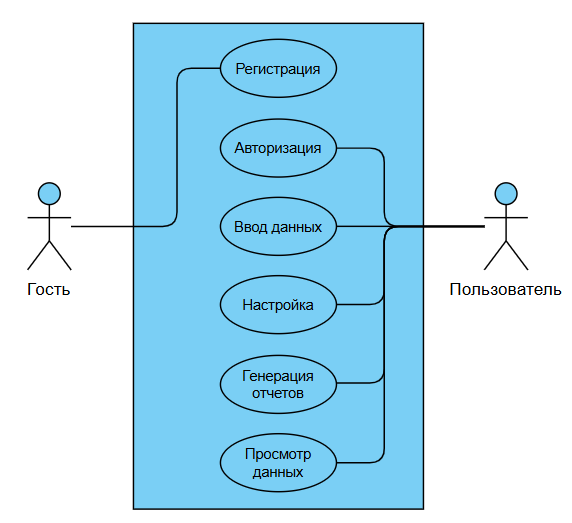


Рисунок 2 — Диаграмма сценариев приложения

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.

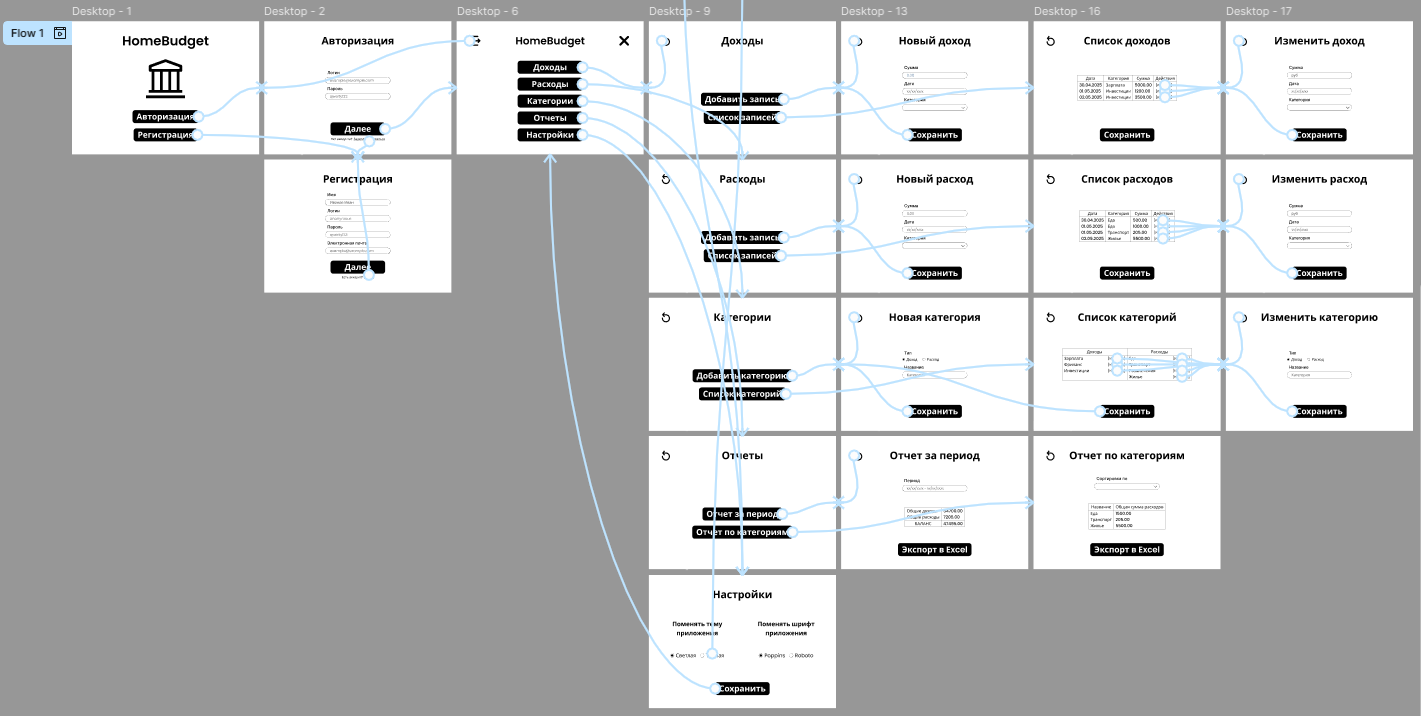


Рисунок 3 — прототип приложения

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. В интерфейсе представлены кнопка возврата на предыдущую страницу, текстовое поле для временного периода, таблица с данными за соответствующий период и кнопка экспорта отчета.

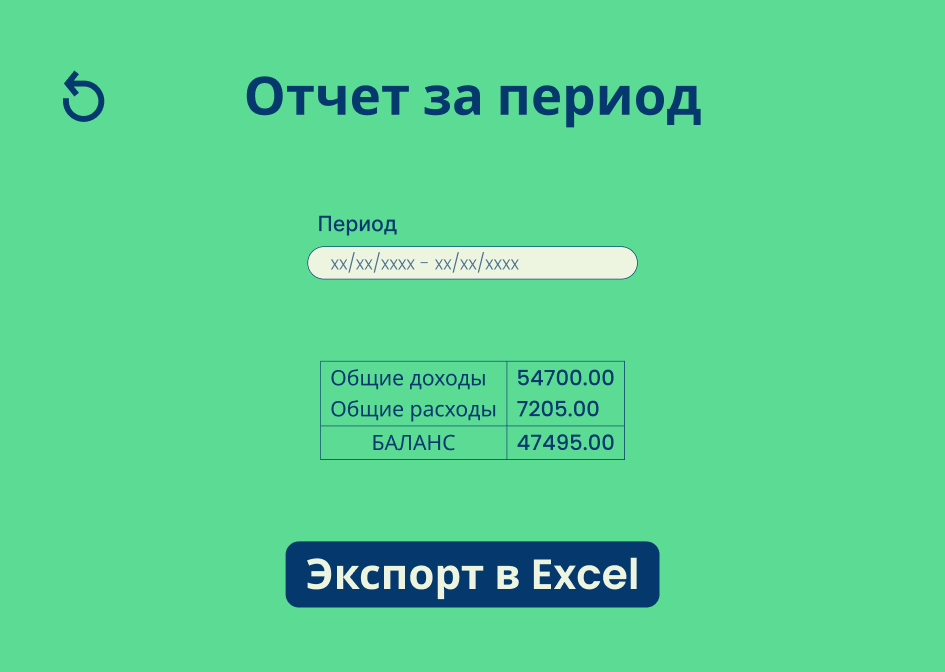


Рисунок 6 — макет страницы создания отчетов

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

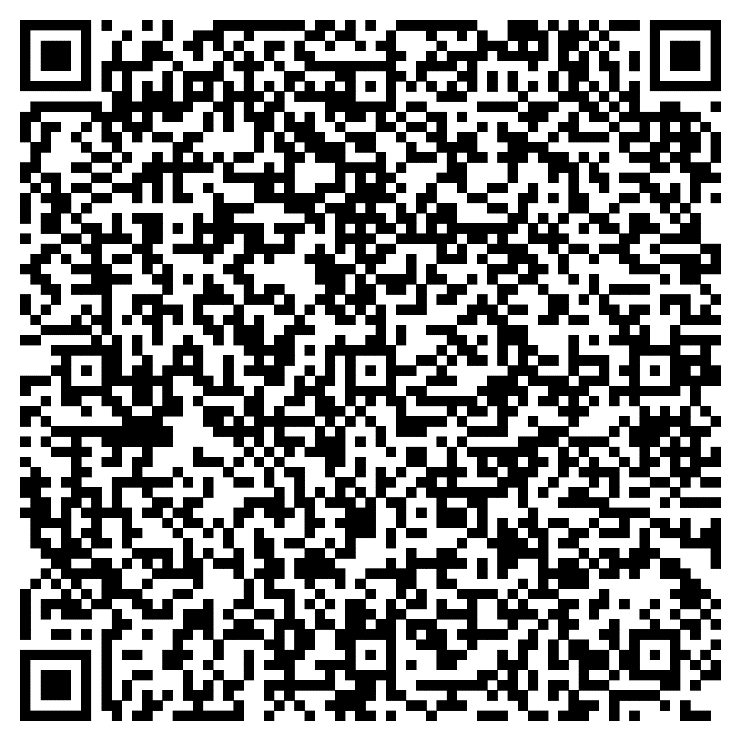


Рисунок 7 — QR-код к прототипу

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).

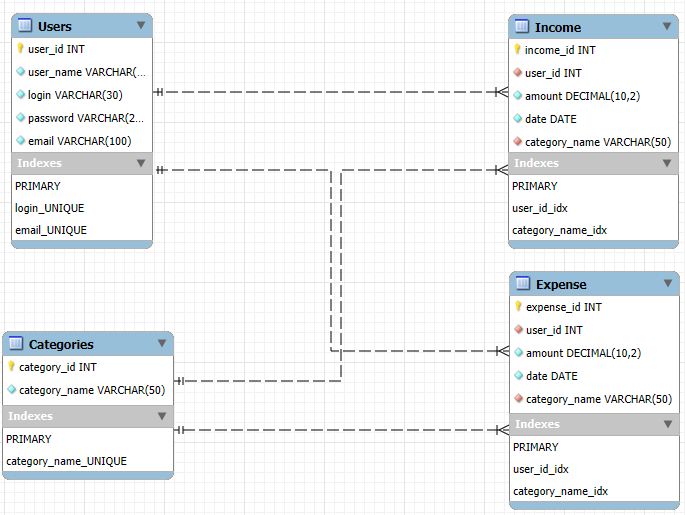


Рисунок 8 – Диаграмма “сущность-связь”

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.

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)