

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Приложение-агрегатор для анализа условий и
подбора ипотечных предложений**

Студент

(подпись, дата)

Голосова Софья Михайловна
(ФИО полностью)

Руководитель

(подпись, дата)

Балашова И.Ю.
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.
(фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«20» декабря 2024 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Голосова Софья Михайловна гр. 21ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Приложение-агрегатор для анализа условий и подбора ипотечных предложений

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

4. Задание на работу (назначение разработки, исходные данные и т. п.)

1) *Назначение: программные средства для анализа условий и подбора ипотечных предложений с возможностью подбора предложений нескольких банков.*

2) *Среда выполнения: Windows 11.*

Языковые средства разработки: C#, SQL, UML.

Инструментальные средства: Visual Studio 2022, MS SQL Server, Visual Paradigm 17.2.

Технологии разработки: ООП, принципы SOLID, шаблоны проектирования.

3) *Основные функции приложения:*

а) регистрация и авторизация;

б) расчет ипотечных параметров;

в) расчет и скачивание графика ежемесячных платежей;

г) подбор банковских предложений по ипотеке;

д) сохранение понравившихся предложений;

е) просмотр имеющихся банковских предложений;

ж) предоставление справочной информации.

5. Перечень подлежащих разработке вопросов

1) *Анализ предметной области и постановка задачи;*

2) *Проектирование;*

3) *Реализация;*

4) *Контроль качества программного обеспечения;*

5) *Оформление пояснительной записки.*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 98 листов, 31 рисунок, 19 таблиц, 21 использованный источник и 5 приложений.

ИПОТЕЧНЫЙ КАЛЬКУЛЯТОР-АГРЕГАТОР, ВЕБ-ПРИЛОЖЕНИЕ, МОНОЛИТНАЯ АРХИТЕКТУРА, MVC, UML, SQL SERVER, C#, RAZOR, БАЗА ДАННЫХ.

Объект исследования – программные средства для расчета параметров ипотеки и подбора соответствующих им банковских предложений.

Цель выпускной квалификационной работы – спроектировать и разработать приложение-агрегатор для анализа условий и подбора ипотечных предложений.

Технологии разработки – разработка средств расчета и подбора ипотеки проводилась в среде разработки Microsoft Visual Studio 2022 на объектно-ориентированном языке программирования C# с использованием языка разметки Razor. Разработка базы данных приложения проводилась в СУБД SQL Server.

Среда исполнения – Windows 11, веб-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были спроектированы и разработаны программные средства для расчета ипотечных параметров и подбора ипотеки. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04 – 08BP211.06 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Приложение-агрегатор для анализа условий и подбора ипотечных предложений. Пояснительная записка			Лит.	Лист	Листов	
Разраб.		Голосова С.М.									
Пров.		Балашова И.Ю.								4	98
Реценз.								Группа 21ВП1			
Н. Контр.		Такташкин Д.В									
Утв.		Козлов А.Ю.									

Содержание

Введение.....	7
1 Анализ предметной области и выявление требований на разработку приложения-агрегатора для анализа условий и подбора ипотечных предложений	8
1.1 Назначение и условия применения калькуляторов ипотечного кредитования	8
1.2 Сравнительный анализ приложений для расчета ипотечных кредитов	8
1.3 Анализ функциональных требований	13
1.4 Анализ нефункциональных требований	16
1.5 Планирование разработки и оценка бюджета	17
2 Проектирование приложения-агрегатора для анализа условий и подбора ипотечных предложений	20
2.1 Архитектура приложения.....	20
2.2 Проектирование структур данных	22
2.2.1 Концептуальная модель базы данных.....	22
2.2.2 Логическая модель базы данных	24
2.3 Проектирование классов программных средств.....	26
2.4 Проектирование пользовательского интерфейса.....	27
3 Реализация приложения-агрегатора для анализа условий и подбора ипотечных предложений	30
3.1 Реализация базы данных.....	30
3.2 Реализация основного приложения.....	34
3.3 Реализация интерфейса приложения	36
4 Контроль качества программного обеспечения.....	47
4.1 Выбор режима тестирования и метрик кода	47
4.2 Тестирование приложения-агрегатора для анализа условий и подбора ипотечных предложений	48

4.2.1 Тестирование пользовательской части приложения для анализа условий и подбора ипотечных предложений.....	48
4.2.2 Тестирование серверной части приложения для анализа условий и подбора ипотечных предложений.....	51
4.3 Оценка качества программного кода	52
4.3.1 Цикломатическая сложность	52
4.3.2 Метрика Холстеда.....	54
4.3.3 Метрика SLOC.....	57
Заключение	58
Список использованных источников	60
Приложение А (справочное) Глоссарий	63
Приложение Б (обязательное) Листинг приложения	65
Приложение В (рекомендуемое) Графическая часть	89
Приложение Г (рекомендуемое) Результаты функционального тестирования.....	92
Приложение Д (рекомендуемое) Листинг модульных тестов.....	96

Введение

Приобретение недвижимости – серьезный шаг для каждого человека. Все чаще в качестве вида оплаты покупки недвижимости выбор падает в сторону ипотечного кредитования. Многие компании-застройщики и ипотечные банки уже полностью перевели в онлайн процессы продаж и оформления ипотеки. Однако перед подачей заявки необходимо изучить банковские предложения, ипотечные условия и многое другое. Для облегчения принятия такого решения созданы платформы – ипотечные калькуляторы.

Разработка приложения-агрегатора для анализа условий и подбора ипотечных предложений является актуальной темой на сегодняшний день, так как такие средства решают проблему ограничения клиента в объективном анализе своих возможностей, помогают наглядно сравнить существующие предложения и подобрать для себя оптимальные условия.

Целью выпускной квалификационной работы является разработка приложения-агрегатора для анализа условий и подбора ипотечных предложений, которое позволяет человеку получить приблизительные расчеты ипотечных параметров, в том числе суммы кредита и переплаты, провести анализ своих финансовых возможностей, и в соответствии с ними подобрать банковские предложения.

Исходя из цели данной работы, были поставлены следующие задачи:

- проанализировать предметную область и существующие аналогичные платформы;
- спроектировать программные средства, а также структуры данных разрабатываемого приложения;
- разработать программные средства;
- осуществить контроль качества программного обеспечения.

Успешное решение поставленных задач позволит создать эффективный инструмент для расчета ипотечных условий.

1 Анализ предметной области и выявление требований на разработку приложения-агрегатора для анализа условий и подбора ипотечных предложений

1.1 Назначение и условия применения калькуляторов ипотечного кредитования

Перед подачей заявки на оформление ипотеки необходимо ознакомиться с актуальными банковскими предложениями и выбрать подходящие условия. С этим могут помочь специализированные платформы – ипотечные калькуляторы – которые зачастую выступают в роли консультантов. Основной целью таких программ является отбор соответствующих предложений и предоставление ориентировочной информации о будущих платежах и размере кредита в удобном виде.

Ипотечный калькулятор – это программа, позволяющая быстро рассчитать размер возможного ипотечного кредита и сумму ежемесячных платежей. Как правило, у каждого банка, предоставляющего услуги ипотечного кредитования, есть свой ипотечный калькулятор. С их помощью можно сэкономить время на сравнении разных ипотечных программ, предоставляемых банком, и выбрать наиболее приемлемый вариант [1].

Большинство существующих ипотечных калькуляторов относятся к конкретным банкам, что вынуждает потенциального заемщика тратить свое время и ресурсы на исследование каждого сайта в отдельности. В связи с этим актуальной задачей является разработка приложения-агрегатора подбора ипотеки, предоставляющего пользователю актуальные сведения из различных источников.

1.2 Сравнительный анализ приложений для расчета ипотечных кредитов

В рамках данного исследования выбраны наиболее востребованные платформы для подбора ипотечного кредитования – «Сравни», «Банки.ру» и

«Циан». Исследование направлено на выявление ключевых достоинств и ограничений указанных сервисов, сопоставление их характеристик.

Приложение «Сравни» является типичным примером приложения для подбора банковских предложений. Интерфейс «Сравни» представлен на рисунке 1.

сравни Кредиты Страхование Займы Ипотека Карты Вклады Бизнес Обучение Журнал

Ипотечный калькулятор

Ипотечный калькулятор - расчет ипотеки с учетом первоначального взноса, процентных ставок и срока кредитования. Доступен выбор 12 предложений в 6 банках с первоначальным взносом от 15 % и процентными ставками от 6% до 29,2%. Используйте универсальный калькулятор ипотеки онлайн для подбора оптимальных условий!

Калькулятор ипотеки

Город покупки недвижимости

Пенза

Тип недвижимости

Квартира в новостройке Квартира на вторичном рынке

Готовый дом / участок Строительство дома Другой

Стоимость недвижимости

4 000 000 Р

Первоначальный взнос ☐ Материнский капитал

1 000 000 Р 25%

20% 25% 30% 35%

Срок ипотеки

20 лет

10 лет 15 лет 20 лет 25 лет 30 лет

Ставка по ипотеке

Ваш расчёт

от 21 302 Р **от 3 000 000 Р**

Платёж в месяц Сумма кредита

Процентная ставка от 5.89%

Полная стоимость кредита от 6.01%

Необходимый доход от 42 605 Р

[Посмотреть график платежей](#)

Предложения под ваши условия

	Альфа-Банк СЕМЕЙНАЯ ИПОТЕКА	Платёж от 21 492 Р	Ставка от 6%	<input type="button" value="Перейти"/>
	Альфа-Банк для IT-специалистов	Платёж от 21 492 Р	Ставка от 6%	<input type="button" value="Перейти"/>
	Совкомбанк ВТОРИЧНОЕ ЖИЛЬЕ	Платёж от 67 800 Р	Ставка от 26.99%	<input type="button" value="Перейти"/>
	Совкомбанк НОВОСТРОЙКА	Платёж от 67 800 Р	Ставка от 26.99%	<input type="button" value="Перейти"/>
	Т-Банк НА ВТОРИЧНОЕ ЖИЛЬЕ	Платёж от 60 522 Р	Ставка от 24%	<input type="button" value="Перейти"/>

[Показать ещё 6](#)

Рисунок 1 – Интерфейс приложения «Сравни»

Ипотечный калькулятор «Сравни» является одним из сервисов экосистемы «Сравни». Широкий выбор банковских предложений является одним из главных преимуществ данной платформы. Графическая интерпретация ежемесячных

платежей позволяет пользователям повысить понимание о соотношении суммы основного долга и процентов по ипотеке.

Интерфейс калькулятора «Сравни» позволяет вводить большое количество параметров для подбора ипотеки, однако это так же является недостатком, поскольку затрудняет использование калькулятора пользователями, не углубленными в сферу ипотечного кредитования. Помимо этого, отсутствует возможность сохранить понравившееся предложение.

Платформа онлайн-сервисов «Банки.ру» позволяет пользователям подбирать и оформлять финансовые продукты по заданным индивидуальным параметрам. Интерфейс ипотечного калькулятора «Банки.ру» представлен на рисунке 2.

The screenshot shows the 'Ипотечный калькулятор' (Mortgage Calculator) interface on the Banki.ru website. The header includes navigation links: Вклады, Кредиты, Займы, Карты, Ипотека, Страхование, Инвестиции, Бизнес, Новости, and Ещё. A search bar and a 'Войти' button are also present.

The main section is titled 'Ипотечный калькулятор' and includes a sub-header 'Ипотека' and 'Рефинансирование'. It features a form with the following fields and values:

- По стоимости** (Selected):
 - Стоимость недвижимости: 4 000 000
 - Первоначальный взнос: 2 500 000
 - Срок в годах: 20
 - Ставка: 10.5
- По платежу** (Unselected):
 - Ежемесячный платеж: 14 976 Р
 - Переплата по кредиту: 2 094 168 Р
 - Рекомендуемый доход: 49 951 Р
- Как накопить на первоначальный взнос?** (Unselected):
 - Сумма кредита: 1 500 000 Р
 - Общая выплата: 3 594 168 Р
 - Налоговый вычет: 532 242 Р

Below the form, there are buttons for 'Заполнить анкету' and 'Смотреть все предложения'. A section titled 'Оставьте заявку на ипотечный кредит по вашим параметрам' shows a list of offers, with the first one from ВТБ (Вторичное жилье) having a full credit cost of 27.849% - 37.070%, a rate of 26.7% - 28.7%, a payment of 33 790 Р, and a down payment of 20.1%.

Рисунок 2 – Интерфейс приложения «Банки.ру»

Наличие возможности скачивания графика платежей отличает ипотечный калькулятор «Банки.ру» от платформ-аналогов и является основным

преимуществом. Удобный ввод основных ипотечных параметров делает платформу доступной для пользователей, которые только начинают знакомство с данным видом кредитования.

Из недостатков стоит отметить отсутствие графической интерпретации расчета ежемесячных платежей, невозможность сохранения понравившегося предложения, а также сложности работы с личным кабинетом, обусловленные необходимостью заполнения специальной анкеты для доступа к профилю.

«Циан» – это популярная онлайн-платформа для покупки, продажи, сдачи в аренду и оценки стоимости недвижимости, а также для расчета и подбора ипотеки. Интерфейс приложения «Циан» представлен на рисунке 3.

Ипотечный калькулятор
С возможностью расчета досрочного погашения

Навигация: Аренда | Продажа | Новостройки | Дома и участки | Коммерческая | Ипотека | Мой дом | Сделка* | Сервисы | Приложение Циан | ПИК

Ваш ежемесячный платеж: **17 565 ₽**

Калькулятор позволяет рассчитать ипотеку с учетом досрочного погашения. Введите параметры:

- Стоимость недвижимости: 3 000 000 ₽
- Первоначальный взнос: 900 000 ₽ (15% - 50%)
- Срок кредита: 20 лет (5 лет - 20 лет)
- Процентная ставка: 8% (2% - 17%)

Результаты расчета:

- Кредит: 2 100 000 ₽
- Проценты: 2 115 658 ₽
- Проценты + Кредит: 4 215 658 ₽
- Необходимый доход: 29 275 ₽

График платежей

Персональные ставки от крупных банков:

- Альфа Банк
- Сбербанк
- МКБ
- СОВКОМБАНК
- Т-БАНК
- Уралсиб
- ВТБ

Рисунок 3 – Интерфейс приложения «Циан»

Среди достоинств можно выделить привычный для кредитных калькуляторов интерфейс, что сокращает время адаптации пользователя в новой

информационной среде, а также возможность подачи заявки на получение ипотеки онлайн.

Основными недостатками приложения являются отсутствие подбора ипотечных предложений по введенным параметрам, и невозможность скачивания графика платежей.

Для сравнения рассмотренных приложений выделен ряд критериев с установленной балльной оценкой:

0 – «полное отсутствие реализации»;

1 – «частичная реализация»;

2 – «полная реализация».

Результаты сравнения программных средств для анализа условий и подбора ипотечных предложений приведены в таблице 1.

Таблица 1 – Сравнительный анализ продуктов-аналогов

Критериальная оценка	Сравни	Банки.ру	Циан
Интуитивно понятный интерфейс	2	2	2
Возможность выбора типа платежа	0	0	0
Возможность получения и скачивания графика платежей	1	2	1
Графическая интерпретация результатов расчета платежей	2	0	0
Подбор предложений разных банков по результатам расчета введенных условий	2	1	0
Возможность сохранить понравившееся предложение	0	0	0
Раздел справочной информации/часто задаваемые вопросы	2	1	1
Итого	9	6	4

Подводя итог сравнения, можно сделать вывод, что указанные аналоги предоставляют схожий функционал, обладают рядом достоинств, но ни одно из рассмотренных приложений не соответствует объявленным критериям в полной мере [1]. Поэтому необходимо разработать приложение-агрегатор, которое будет предоставлять пользователю следующий функционал:

– удобный ввод ипотечных условий;

- выбор предложений из нескольких банков;
- сохранение в личном кабинете понравившегося предложения;
- скачивание графика платежей;
- доступность справочной информации.

Исходя из определенных выше требований к специализированным платформам, принято решение разработать приложение-агрегатор для анализа условий и подбора ипотечных предложений. Новый ипотечный калькулятор будет включать функции расчета основных ипотечных параметров, получения графика платежей, подбора актуальных банковских предложений по введенным параметрам, возможность сохранения понравившегося предложения, а также справочную информацию для лучшего понимания процесса и результатов расчета ипотеки, что делает его инструментом с более широкими возможностями функционала для расчета ипотечных параметров. Использование подобного агрегатора играет важную роль в упрощении поиска, а также доступа к необходимым данным.

1.3 Анализ функциональных требований

В процессе анализа существующих ипотечных калькуляторов были выделены следующие функциональные требования:

- регистрация и авторизация пользователя;
- редактирование данных для входа пользователя;
- выход из аккаунта пользователя;
- расчет ипотечных параметров;
- подбор предложений по ипотечным параметрам;
- скачивание графика платежей;
- просмотр имеющихся предложений;
- добавление выбранного предложения в «избранное» пользователя;
- удаление предложения из «избранного» пользователя;
- создание заметки в «избранном» пользователя;
- получение справочной информации [2].

Диаграмма вариантов использования в унифицированном языке моделирования (UML) – это визуальное представление, которое иллюстрирует взаимодействие между пользователями (актерами) и системой [3].

В соответствии с выявленными требованиями разработана диаграмма вариантов использования UML, представленная на рисунке 4.



Рисунок 4 – Диаграмма вариантов использования

Диаграмма отражает функциональные требования, а также дает общее представление о поведении системы. Основные варианты использования системы подразумевают работу с калькулятором, результатами расчета и личным кабинетом пользователя. Сценарий варианта использования «Рассчитать условия ипотеки» представлен на рисунке 5.

Прецедент: Рассчитать условия ипотеки
ID: 1
Краткое описание: Пользователь выполняет расчет в ипотечном калькуляторе-агрегаторе
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь перешел на страницу калькулятора
Основной поток: 1. Пользователь вводит ипотечные параметры 2. Пользователь получает расчет ипотечных параметров и график платежей 3. Пользователь просматривает подобранные предложения
Постусловие: 1. Пользователь получил расчет ипотечного калькулятора

Рисунок 5 – Сценарий варианта использования «Рассчитать условия ипотеки»

Сценарий варианта использования «Войти в аккаунт» представлен на рисунке 6.

Прецедент: Войти в аккаунт
ID: 2
Краткое описание: Пользователь регистрируется или авторизуется в приложении
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь перешел на страницу личного кабинета в калькуляторе 2. Пользователь не был зарегистрирован/авторизован
Основной поток: 1. Пользователь вводит данные для входа в аккаунт (логин и пароль) 2. Пользователь получает доступ к расширенному функционалу приложения Альтернативные потоки: 2а. Пользователь вводит данные для регистрации (логин и пароль)
Постусловие: 1. Пользователь вошел в приложение 2. Пользователю доступен расширенный функционал приложения

Рисунок 6 – Сценарий варианта использования «Войти в аккаунт»

Сценарий варианта использования «Работать с «Избранным»» представлен на рисунке 7.

Прецедент: Работать с «Избранным»
ID: 3
Краткое описание: Пользователь просматривает предложения в «Избранном»
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь авторизован 2. Пользователь перешел на страницу «Избранное»
Основной поток: 1. Пользователь просматривает сохраненные предложения Альтернативные потоки: 2а. Пользователь удаляет сохраненное предложение 2б. Пользователь создает заметку
Постусловие: 1. Пользователь отредактировал «Избранное»

Рисунок 7 – Сценарий варианта использования «Работать с «Избранным»»

Проведенный анализ функциональных требований позволил выделить варианты использования приложения-агрегатора для анализа условий и подбора ипотечных предложений, что в свою очередь предоставляет возможность увидеть пути взаимодействия пользователя с системой. Представленные спецификации некоторых из основных бизнес-процессов более подробно отображают выполняемые манипуляции пользователя и приложения.

1.4 Анализ нефункциональных требований

Разработка приложения-агрегатора для анализа условий и подбора ипотечных предложений должна быть выполнена в многофункциональной среде разработки Microsoft Visual Studio 2022 с применением шаблона проектирования ASP.NET Core MVC на объектно-ориентированном языке программирования C#. Графический интерфейс должен быть построен с использованием языка C# в Razor разметке.

В качестве СУБД должен выступать Microsoft SQL Server. Взаимодействие с СУБД необходимо разработать с использованием фреймворка EntityFramework 9. Приложение должно разворачиваться и исполняться на операционных системах семейства Windows версии 10 и выше.

Приложение должно быть оптимизировано для работы в современных браузерах, таких как Google Chrome (версии 80 и выше), Mozilla Firefox (версии 75 и выше), Apple Safari (версии 13 и выше), Microsoft Edge (версии 80 и выше).

1.5 Планирование разработки и оценка бюджета

При разработке программного продукта необходимо определить задачи проекта, установить сроки и приоритеты для эффективного управления процессом, а также оценить бюджет проекта.

Для управления проектом была выбрана программа Microsoft Office Project 2007, в которую были внесены данные об этапах создания приложения. Краткое описание выделенных задач представлено на рисунке 8.

Название задачи	Длительность	Начало
Начало проекта	0 дней	Вт 08.04.25
Анализ предметной области и выявление требований на разработку приложения-агрегатора	5 дней	Вт 08.04.25
Анализ назначения и условий применения объектов предметной области	1 день	Вт 08.04.25
Анализ приложений для расчета ипотечных кредитов	1 день	Ср 09.04.25
Анализ функциональных требований	1 день	Чт 10.04.25
Анализ нефункциональных требований	1 день	Пт 11.04.25
Планирование процесса разработки приложения и оценка затрат на разработку	1 день	Пн 14.04.25
Анализ предметной области и выявление требований на разработку приложения-агрегатора	0 дней	Пн 14.04.25
Проектирование приложения-агрегатора для анализа условий и подбора ипотечных предложений	9 дней	Вт 15.04.25
Проектирование архитектуры приложения	3 дней	Вт 15.04.25
Проектирование структур данных	2 дней	Пт 18.04.25
Проектирование пользовательского интерфейса	2 дней	Вт 22.04.25
Проектирование классов программных средств	2 дней	Чт 24.04.25
Проектирование приложения-агрегатора для анализа условий и подбора ипотечных предложений	0 дней	Пт 25.04.25
Реализация приложения-агрегатора для анализа условий и подбора ипотечных предложений	13 дней	Пн 28.04.25
Реализация базы данных	2 дней	Пн 28.04.25
Реализация основного приложения	6 дней	Ср 30.04.25
Реализация интерфейса приложения	5 дней	Чт 08.05.25
Реализация приложения-агрегатора для анализа условий и подбора ипотечных предложений	0 дней	Ср 14.05.25
Контроль качества программного обеспечения	6 дней	Чт 15.05.25
Выбор режима тестирования и метрик кода	1 день	Чт 15.05.25
Тестирование пользовательской части приложения для анализа условий и подбора ипотечных предложений	2 дней	Пт 16.05.25
Тестирование серверной части приложения для анализа условий и подбора ипотечных предложений	2 дней	Вт 20.05.25
Оценка качества программного кода	1 день	Чт 22.05.25
Контроль качества программного обеспечения завершен	0 дней	Чт 22.05.25
Оформление отчета	8 дней	Пт 23.05.25
Оформление пояснительной записки	8 дней	Пт 23.05.25
Оформление отчета завершенно	0 дней	Вт 03.06.25
Конец проекта	0 дней	Вт 03.06.25

Рисунок 8 – Сведения об этапах проекта

Для определения длительности проекта в целом и каждой из его фаз служит календарный план. Графическая интерпретация календарного плана представлена диаграммой Ганта (рисунок 9)



Рисунок 9 – Распределение ресурсов на диаграмме Ганта

В соответствии с полученным заданием за дату начала проекта взято 8 апреля 2025 года, по полученной диаграмме Ганта видно, что завершение проекта планируется 3 июня 2025 года.

Для оценки финансовых затрат на реализацию приложения ресурсам были назначены ставки. Ставка каждого используемого ресурса представлена на рисунке 10.

Название ресурса	Тип	Краткое название	Стандартная ставка	Ставка сверхурочных
Аналитик	Трудовой	А	200,00 Р/ч	250,00 Р/ч
Программист	Трудовой	П	300,00 Р/ч	400,00 Р/ч
Дизайнер	Трудовой	Д	200,00 Р/ч	250,00 Р/ч
Разработчик базы данных	Трудовой	Р	250,00 Р/ч	300,00 Р/ч
Тестировщик	Трудовой	Т	200,00 Р/ч	250,00 Р/ч
Компьютер	Материальный	К	70 000,00 Р	
Электричество	Затраты	Э		
Интернет	Затраты	И		

Рисунок 10 – Ресурсы проекта

Для получения оценки занятости трудовых ресурсов, а также оценки бюджета проекта были сгенерированы соответствующие отчеты в программе Microsoft Office Project 2007. Отчет об использовании ресурсов представлен на рисунке В.1 в приложении В. Отчет о бюджете проекта представлен на рисунке В.2 в приложении В.

Таким образом, по отчету об использовании трудовых ресурсов и отчету об общих затратах проекта можно сделать вывод, что для разработки приложения-агрегатора для анализа условий и подбора ипотечных предложений необходимо 98630 рублей, а общая занятость трудовых ресурсов составляет 392 часа с 08.04.2025 по 03.06.2025, соответственно 41 день.

2 Проектирование приложения-агрегатора для анализа условий и подбора ипотечных предложений

2.1 Архитектура приложения

Разработка архитектуры приложения представляет собой процесс проектирования структуры, компонентов и взаимодействия всех частей, обрабатывающих данные, выполняющих бизнес-логику и обеспечивающих взаимодействие с клиентом и другими системами. При разработке архитектуры важно учитывать масштабируемость, технологии, безопасность и производительность приложения, чтобы избежать основных трудностей, возникающих в процессе разработки [4].

Монолитная архитектура – это методология проектирования программного обеспечения, которая объединяет все компоненты приложения в единое неразделимое целое. При такой архитектуре пользовательский интерфейс, бизнес-логика и уровни доступа к данным создаются, используются и поддерживаются как единое целое. Ключевыми компонентами монолитной архитектуры являются:

- пользовательский интерфейс (UI);
- логика приложения;
- уровень доступа к данным;
- база данных [5].

Компонентами приложения с шаблоном проектирования MVC являются модель, представление и контроллер. Контроллер – это компонент, который обрабатывает входящий запрос. Компоненты контроллера выполняют несколько действий для обработки запроса. Компонент контроллера создаёт модель, необходимую для представления. Модель – это компонент, который в основном содержит классы, используемые для хранения данных предметной области или, можно сказать, бизнес-данных. В шаблоне проектирования MVC компонент модели также содержит необходимую логику для получения данных из базы данных.

Как только контроллер создает модель, он выбирает представление для отображения данных домена или модели. При выборе представления контроллер также передает данные модели. В шаблоне проектирования MVC единственной обязанностью представления является отображение данных модели. Таким образом, в MVC представление – это компонент, отвечающий за создание необходимого HTML-кода для отображения данных модели. Как только представление создает HTML-код, этот HTML-код отправляется клиенту по сети, который изначально отправил запрос [6].

Согласно выделенным требованиям к программным средствам, была разработана диаграмма развертывания, отражающая архитектуру приложения-агрегатора для анализа условий и подбора ипотечных предложений. Разработанная диаграмма представлена на рисунке 11.

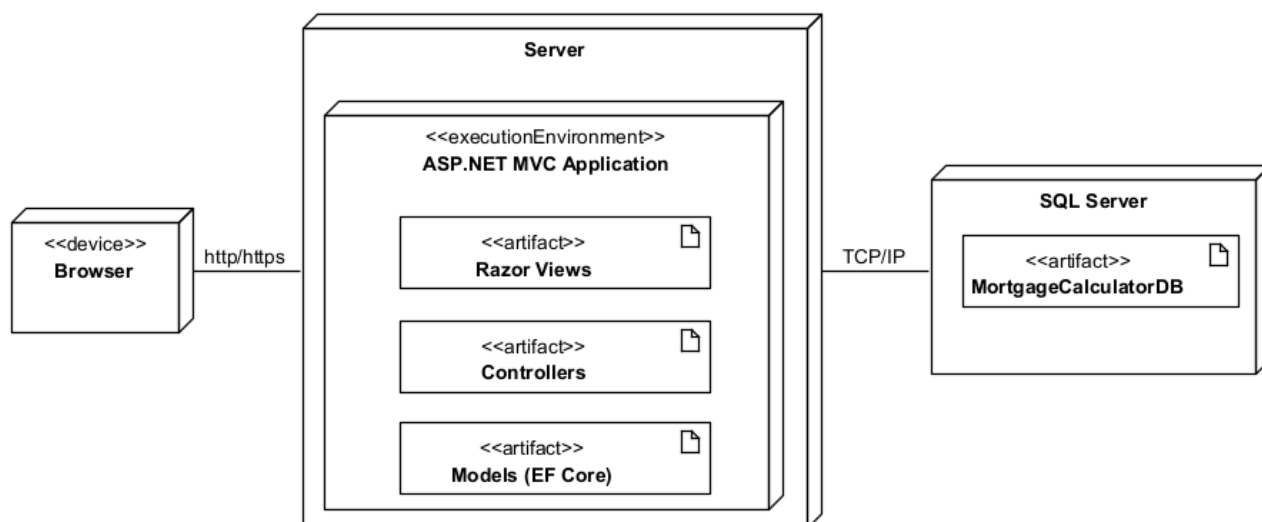


Рисунок 11 – Архитектура приложения-агрегатора для анализа условий и подбора ипотечных предложений

Серверная часть представлена в виде трех основных компонентов шаблона MVC, что позволяет вносить изменения в один из компонентов, оказывая минимальное воздействие на другие. Сервер обрабатывает поступивший от пользователя запрос, обращается к базе данных, анализирует полученные данные и после отображает результат обработки в обновленном представлении.

Четкое выделение компонентов увеличивает возможность масштабирования системы, обеспечивая эффективную работу каждого элемента.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Концептуальная модель служит основой для сбора требований перед проектированием базы данных, используется для определения структуры базы данных на начальном этапе проектирования. Концептуальная модель данных описывает бизнес-процессы, связанные с данными, без указания технических деталей реализации. Хорошо продуманная концептуальная модель данных эффективно представляет базу данных в реальном контексте [7].

Перед составлением концептуальной модели базы данных необходимо выделить основные сущности предметной области разрабатываемого приложения-агрегатора для анализа условий и подбора ипотечных предложений. Табличное описание сущностей представлено в таблице 2.

Таблица 2 – Табличное описание сущностей для концептуальной модели базы данных

Название сущности	Идентификатор сущности	Назначение сущности
Пользователь	AspNetUsers	Содержит сведения о зарегистрированном пользователе калькулятора
Избранное	DbFavorite	Содержит сведения о сохраненных пользователем предложениях и созданных комментариях
Банковское предложение	DbOffer	Содержит сведения о банковском предложении
Банки	DbBank	Содержит сведения о банке
Состояние	DbCondition	Содержит сведения об актуальности банковского предложения
Дата состояния	DbDataCondition	Содержит сведения о дате начала и окончания действия банковского предложения
Тип платежа	DbPaymentType	Содержит сведения о типах расчета ежемесячных платежей

Графическое представление концептуальной модели представлено на рисунке 12.

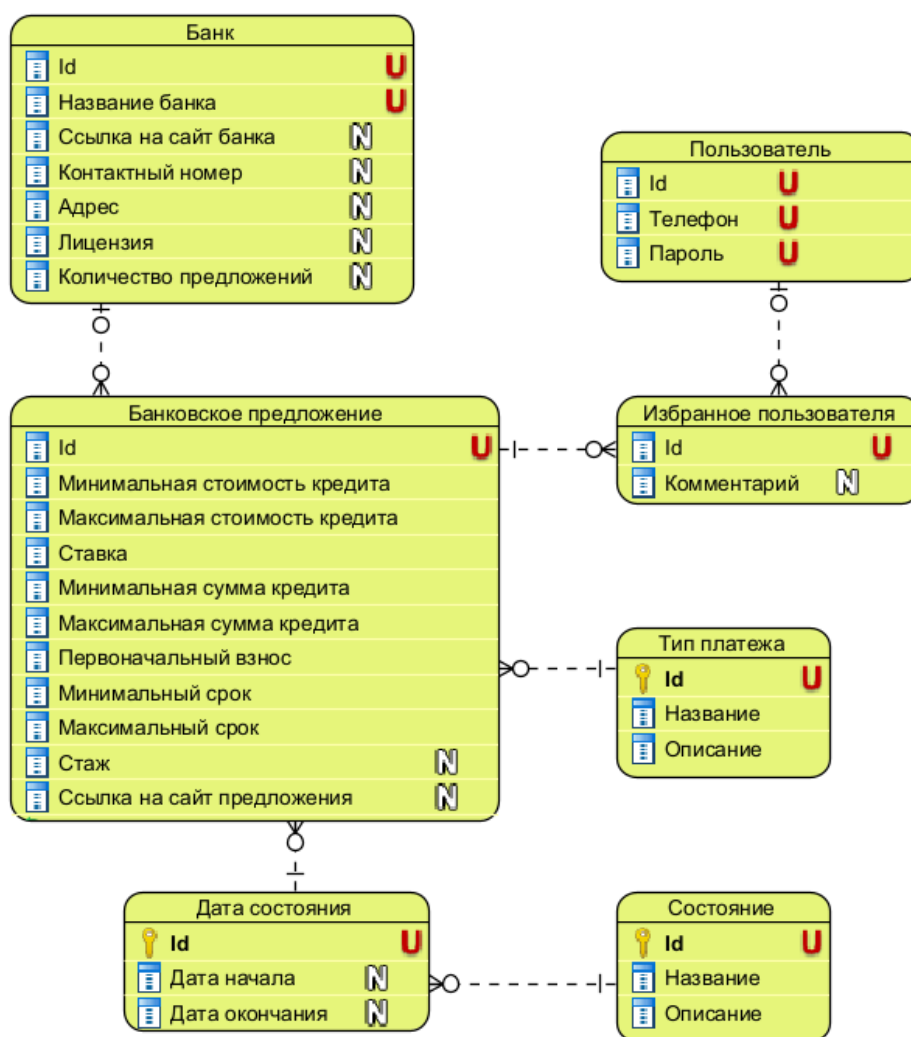


Рисунок 12 – Диаграмма сущность-связь концептуальной модели

Составлено табличное описание связей между сущностями, представленное в таблице 3.

Таблица 3 – Связи между сущностями

Связь	Сущность 1	Сущность 2	Тип связи	Описание
Пользователь – Избранное пользователя	Пользователь	Избранное пользователя	1:N	Один пользователь может иметь несколько избранных предложений.
Банковское предложение – Избранное пользователя	Банковское предложение	Избранное пользователя	1:N	Одно банковское предложение относится к нескольким пользовательским избранным
Банк – Банковское предложение	Банк	Банковское предложение	1:N	Один банк может иметь несколько банковских предложений

Продолжение таблицы 3

Связь	Сущность 1	Сущность 2	Тип связи	Описание
Тип платежа – Банковское предложение	Тип платежа	Банковское предложение	1:N	Один тип платежа может относиться к нескольким банковским предложениям
Состояние – Дата состояния	Состояние	Дата состояния	1:N	Одно состояние может относиться к нескольким датам
Дата состояния – Банковское предложение	Дата состояния	Банковское предложение	1:N	Одна дата может относиться к нескольким предложениям

Концептуальная модель данных построено в среде Visual Paradigm 17.0 в нотации IDEF1X.

2.2.2 Логическая модель базы данных

Логическая модель данных – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила [8]. Логическая модель является следующим этапом при проектировании базы данных, она сложнее концептуальной модели, так как возникает необходимость определить типы столбцов каждой сущности.

Концептуальная модель данных строится на основе нотации сущность-связь, включающей описание сущностей, атрибутов и связей между сущностями. В ходе логического проектирования реляционной базы данных (РБД) элементы концептуальной модели «сущность-связь» отображаются в элементы реляционной модели (отношения, атрибуты отношений, связи между отношениями).

Наиболее часто на этапе логического проектирования РБД применяется предметно-ориентированный подход, при котором схема базы данных строится для конкретной предметной области с учетом правил логического проектирования. При этом сущностям концептуальной модели «сущность-связь» в реляционной модели ставится в соответствие отношение, каждому

свойству сущности – атрибут отношения, связи между сущностями – связь между отношениями.

На основе разработанной концептуальной модели была построена логическая модель базы данных, представленная на рисунке 13.

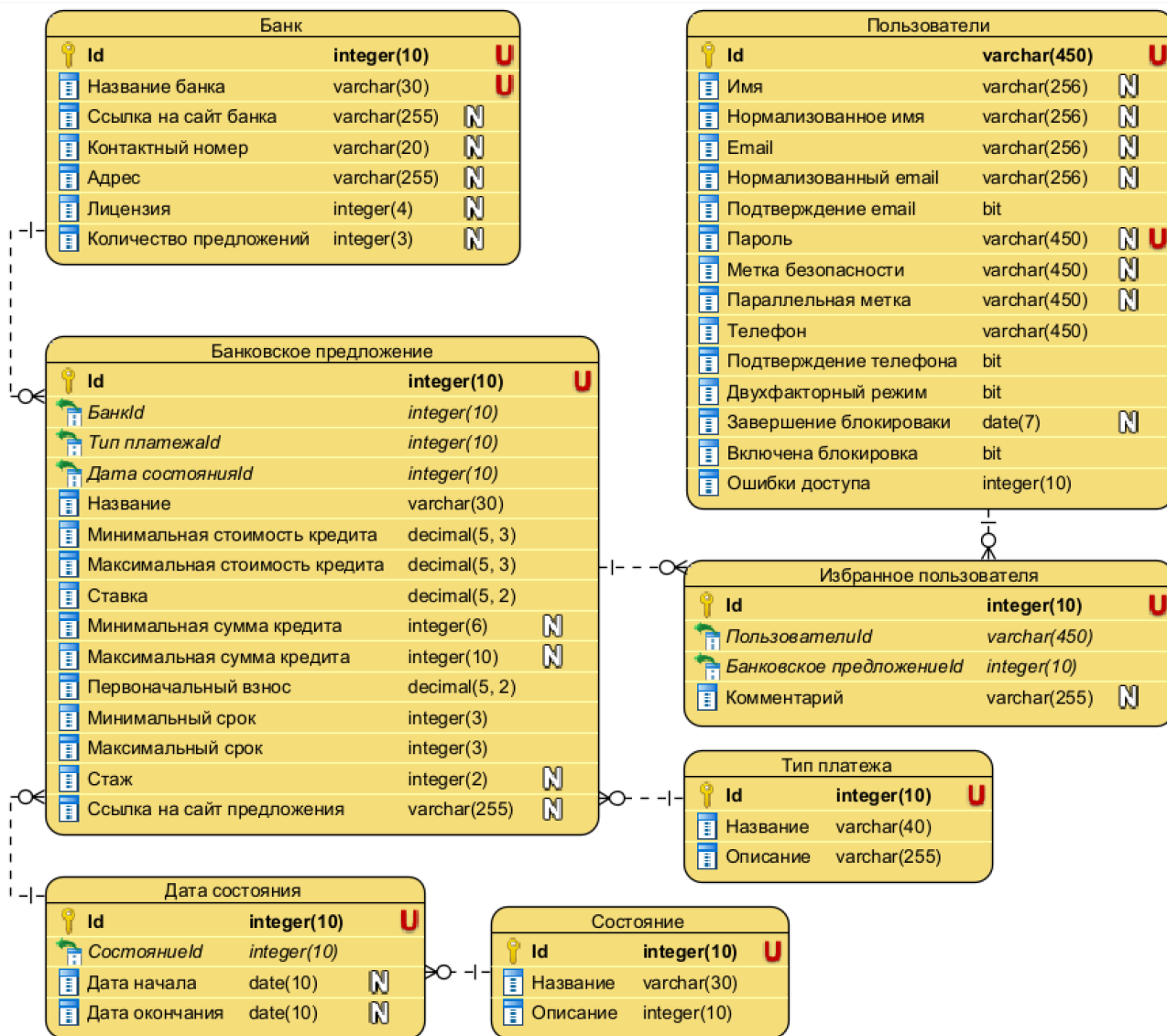


Рисунок 13 – Диаграмма сущность-связь логической модели

Таким образом, спроектированное графическое представление логической модели позволяет перейти к этапу реализации запросов к базе данных приложения, а также наглядно отразить структурные связи между данными.

2.3 Проектирование классов программных средств

В соответствии с выбранной архитектурой приложения-агрегатора для анализа условий и подбора ипотечных предложений была спроектирована диаграмма классов, представленная на рисунке 14.

Диаграммы классов – это разновидность диаграмм унифицированного языка моделирования, используемых в разработке программного обеспечения для визуального представления структуры и взаимосвязей классов в системе [9].

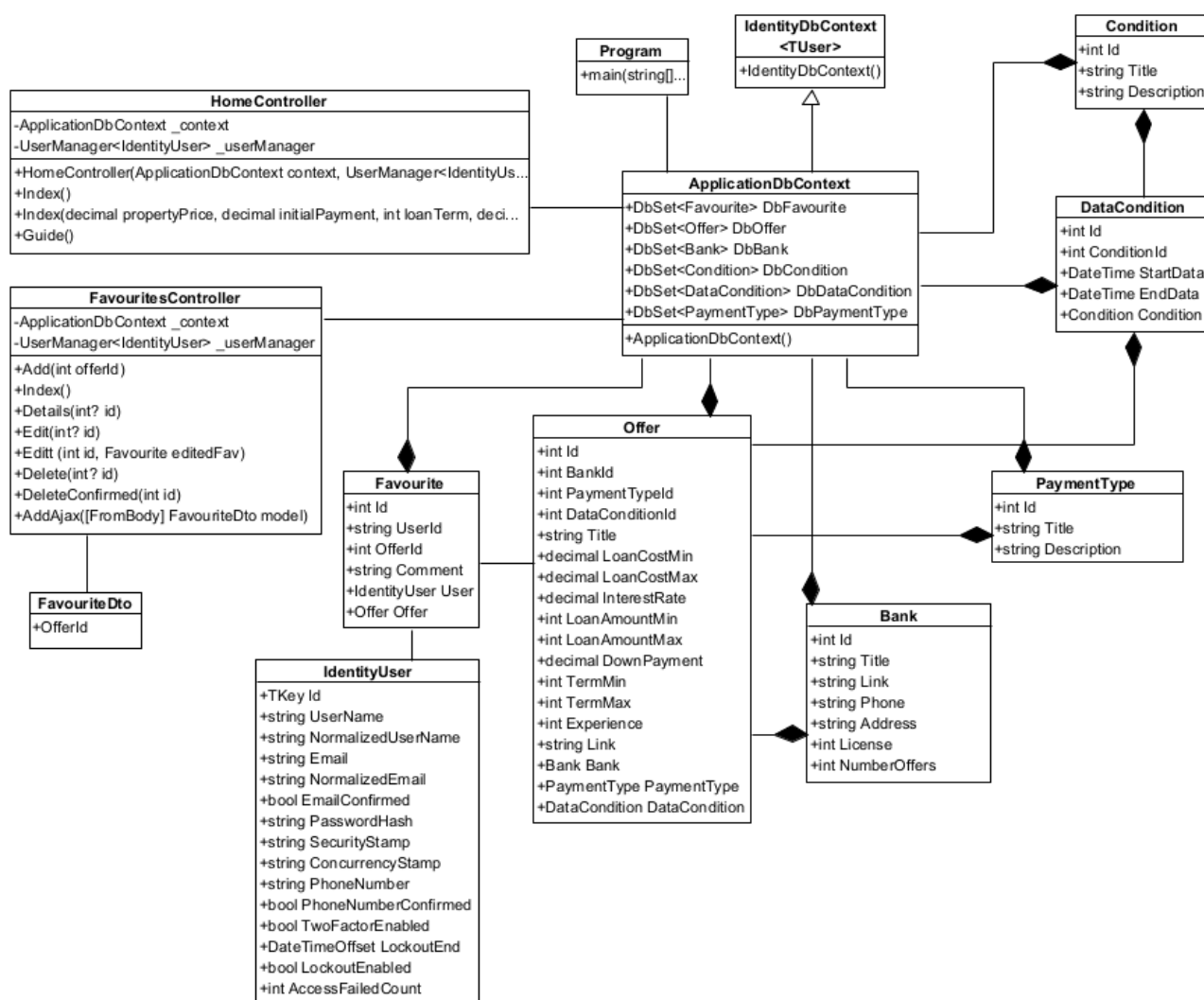


Рисунок 14 – Диаграмма классов

Краткое описание функционального назначения наиболее важных классов разрабатываемой системы приведено в таблице 4.

Таблица 4 – Компоненты диаграммы классов

Компонент	Назначение
Program	Точка входа в программу
ApplicationDbContext	Класс, отвечающий за подключение к базе данных, конфигурацию и доступ к таблицам через DbSet.
IdentityDbContext	Базовый контекст базы данных для работы с системой аутентификации и авторизации в ASP.NET Identity
HomeController	Класс, реализующий обработку пользовательских запросов на главной странице, рассчитывает ипотечные параметры и отображает результат пользователю
FavouritesController	Класс, реализующий управление избранными предложениями пользователя: добавление, удаление, редактирование и отображение списка избранного
Favourite	Модель, представляющая избранное предложение пользователя
Offer	Модель, представляющая ипотечные предложения от банка
Bank	Модель, описывающая название и контактную информацию банка
PaymentType	Модель, описывающая тип расчета ежемесячных платежей по ипотеке
DataCondition	Модель, описывает временные рамки действия ипотечного предложения
Condition	Модель, описывающая актуальность действия предложения
FavouriteDto	Модель, для реализации передачи данных при добавлении избранного через AJAX
IdentityUser	Встроенный класс ASP.NET Identity, представляющий пользователя системы

Спроектированная диаграмма классов наглядно отображает структуру программы, тем самым играет важную роль в жизненном цикле разработки программного обеспечения.

2.4 Проектирование пользовательского интерфейса

Для реализации продукта, способного заинтересовать большее количество пользователей, необходимо спроектировать информативный и удобный дизайн пользовательского интерфейса. Дизайн пользовательского интерфейса – это

часть процесса разработки программного обеспечения, которая создаётся после глубокого изучения пользователей и особенностей предметной области [10].

В соответствии с результатами анализа предметной области и выявленными требованиями к разрабатываемому продукту был спроектирован пользовательский интерфейс.

Интерфейс веб-приложения должен быть понятным и удобным в использовании для пользователей с разным уровнем вовлеченности в предметную область, что позволит увеличить количество потенциальных пользователей.

Проектируемый интерфейс включает в себя несколько страниц:

- главная страница калькулятора;
- страница справочной информации;
- страница регистрации в приложении;
- страница входа в приложение;
- страница избранных предложений пользователя;
- страница управления аккаунтом.

Страницы избранного и управления аккаунтом доступны только для авторизованных пользователей.

Каждая страница сопровождается навигационным меню по приложению в верхней части. Данное расположение является типичным для данной предметной области, что позволяет ускорить адаптацию новых пользователей. На главной странице должен располагаться ипотечный калькулятор, после взаимодействия с которым пользователь сможет увидеть результаты расчета, график платежей, а также подобранные по введенным условиям ипотечные предложения.

Страница справочной информации должна предоставлять информацию о возможностях использования калькулятора, а также о сложных терминах предметной области, использующихся при расчете. Избранное пользователя должно содержать сохраненные предложения, заметки к предложениям, а также

навигацию для перехода на страницы редактирования заметок, удаления предложения и страницы с подробной информацией о предложении.

Интерфейс будет реализован с использованием представлений, которые являются неотъемлемой частью шаблона проектирования MVC. Представления будут созданы с помощью лаконичного и эффективного языка разметки Razor. В Razor шаблон представления сочетает в себе HTML-страницу с несколькими фрагментами кода. Код во фрагментах, написанный на языке C#, вычисляется при рендеринге представления, и полученная разметка интегрируется в шаблон HTML. Механизм Razor считывает шаблоны представлений из физического местоположения на диске. Путь определяется с помощью поставщика виртуальных путей ASP.NET [11, 12].

3 Реализация приложения-агрегатора для анализа условий и подбора ипотечных предложений

3.1 Реализация базы данных

Физическая ER-диаграмма представляет собой реальный проект реляционной базы данных. Она показывает, как данные должны быть структурированы и связаны в конкретной СУБД, поэтому при разработке физической ER-диаграммы важно учитывать соглашения и ограничения используемой СУБД. Это означает, что для столбцов сущностей необходимо точно использовать тип данных, а при именовании сущностей и столбцов следует избегать использования зарезервированных слов. Кроме того, становится возможным добавление первичных и внешних ключей, а также ограничений [13].

Для построения физической модели базы данных приложения использовалась среда Visual Paradigm 17.2, нотация IDEF1X. Разработанная модель представлена на рисунке 15.

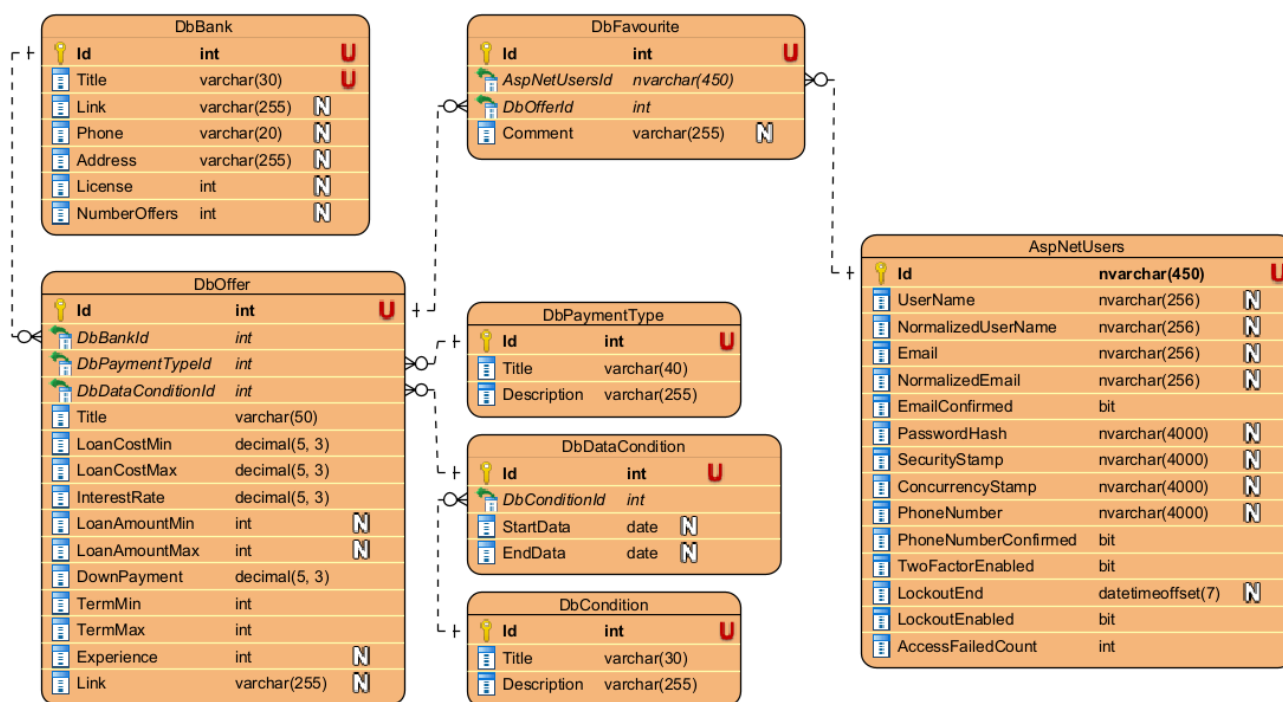


Рисунок 15 – Диаграмма сущность-связь физической модели

Описание атрибутов сущности «Банк» приведено в таблице 5.

Таблица 5 – Сущность «Банк»

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Идентификатор пользователя	Id	integer	PK		Да
Название банка	Title	varchar	-	30	Да
Ссылка на сайт банка	Link	varchar	-	255	Нет
Контактный телефон	Phone	varchar	-	20	Нет
Адрес	Address	varchar	-	255	Нет
Лицензия (регистрационный номер)	License	integer	-	5	Нет
Количество предложений	NumberOffers	integer	-	3	Нет

Описание атрибутов сущности «Банковское предложение» приведено в таблице 6.

Таблица 6 – Сущность «Банковское предложение»

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Идентификатор предложения	Id	int	PK		Да
Идентификатор банка	DbBankId	integer	FK		Да
Идентификатор типа платежа	DbPaymentTypeId	integer	FK		Да
Идентификатор даты состояния платежа	DbDataConditionId	integer	FK		Да
Название предложения	Title	varchar	-	50	Да
Минимальная полная стоимость кредита	LoanCostMin	decimal	-	5,3	Да
Максимальная полная стоимость кредита	LoanCostMax	decimal	-	5,3	Да
Процентная ставка	InterestRate	decimal	-	5,3	Да
Минимальная сумма кредита	LoanAmountMin	integer	-	6	Нет
Максимальная сумма кредита	LoanAmountMax	integer	-	10	Нет
Первоначальный взнос	DownPayment	decimal	-	5,3	Да
Минимальный срок ипотеки в месяцах	TermMin	integer	-	3	Да

Продолжение таблицы 6

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Максимальный срок ипотеки в месяцах	TermMax	integer	-	3	Да
Стаж	Experience	integer	-	3	Нет
Ссылка на сайт предложения	Link	varchar	-	255	Нет

Описание атрибутов сущности «Тип платежа» приведено в таблице 7.

Таблица 7 – Сущность «Тип платежа»

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Идентификатор типа платежа	Id	integer	PK		Да
Название	Title	varchar	-	40	Да
Описание	Descriptions	varchar	-	255	Да

Описание атрибутов сущности «Сроки состояния» приведено в таблице 8.

Таблица 8 – Сущность «Сроки состояния»

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Идентификатор даты состояния	Id	integer	PK		Да
Идентификатор состояния	DbConditionId	integer	FK		Да
Дата начала действия предложения	StartDate	data	-		Нет
Дата окончания действия предложения	EndDate	data	-		Нет

Описание атрибутов сущности «Состояние» приведено в таблице 9.

Таблица 9 – Сущность «Состояние»

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Идентификатор состояния	Id	integer	PK		Да
Название	Title	varchar	-	30	Да
Описание	Descriptions	varchar	-	255	Да

Описание атрибутов сущности «Пользователь» приведено в таблице 10.

Таблица 10 – Сущность «Пользователь»

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Идентификатор пользователя	Id	nvarchar	PK		Да
Имя пользователя	UserName	nvarchar	-	256	Нет
Нормализованное имя пользователя	NormalizedUserName	nvarchar	-	256	Нет
Адрес электронной почты пользователя	Email	nvarchar	-	256	Нет
Нормализованный адрес электронной почты пользователя	NormalizedEmail	nvarchar	-	256	Нет
Флаг подтверждения электронной почты	EmailConfirmed	bit	-		Да
Хэш-представление пароля пользователя	PasswordHash	nvarchar	-	4000	Нет
Случайное значение, обновляющееся при изменении учетных данных пользователя	SecurityStamp	nvarchar	-	4000	Нет
Случайное значение, обновляющееся при сохранении пользователя в хранилище	ConcurrencyStamp	nvarchar	-	4000	Нет
Номер телефона пользователя	PhoneNumber	nvarchar	-	4000	Нет
Флаг подтверждения телефона пользователя	PhoneNumberConfirmed	bit	-		Да
Флаг, включена ли двухфакторная проверка	TwoFactorEnabled	bit	-		Да
Дата и время в формате UTC завершения блокировки пользователя	LockoutEnd	datetimeoffset	-		Нет
Флаг, может ли пользователь быть заблокирован	LockoutEnabled	bit	-		Да
Количество неудачных попыток входа	AccessFailedCount	int	-		Да

Описание атрибутов сущности «Избранное» приведено в таблице 11.

Таблица 11 – Сущность «Избранное»

Содержание поля	Системное наименование параметров таблицы	Тип	Ключ	Размер	Обязательность
Идентификатор избранного	Id	integer	PK		Да
Идентификатор пользователя	AspNetUsersId	integer	FK		Да
Идентификатор предложения	DbOfferId	integer	FK		Да
Комментарий	Comment	varchar	-	255	Нет

База данных, соответствующая разработанной физической модели, была развернута с помощью механизма миграций. Миграции – это функция в Entity Framework, которая позволяет управлять изменениями в схеме базы данных с течением времени. Загрузка данных позволяет предварительно заполнить базу данных набором исходных данных [14]. Для работы с базой данных использовалась среда Microsoft SQL Server Management Studio.

3.2 Реализация основного приложения

Реализация основного приложения проводилась в среде разработки Microsoft Visual Studio 2022, на языке программирования C#, с использованием среды ASP.NET Core и шаблона MVC.

ASP.NET Core является кроссплатформенной средой с открытым исходным кодом для создания современных приложений, оптимизированных для работы в облаке. Приложения ASP.NET Core отличаются простотой и модульностью, а также реализуют встроенную поддержку внедрения зависимостей, что позволяет расширить возможности тестирования и сопровождения. В сочетании с моделью MVC, которая поддерживает создание современных веб-API в дополнение к приложениям на основе представлений, ASP.NET Core представляет собой эффективную платформу для разработки веб-приложений [15].

Общая архитектура приложения использует множество различных представлений MVC и конечных точек URL-адресов. Такой подход позволяет минимизировать количество функций, обрабатываемых на стороне клиента, тем

самым разделяя ответственность между сервером и клиентом. При этом отдельные операции на конкретной странице оптимизированы с использованием функций AJAX (асинхронный JavaScript и XML). Для наглядного отображения взаимосвязи между компонентами системы была разработана диаграмма компонентов, представленная на рисунке 16.

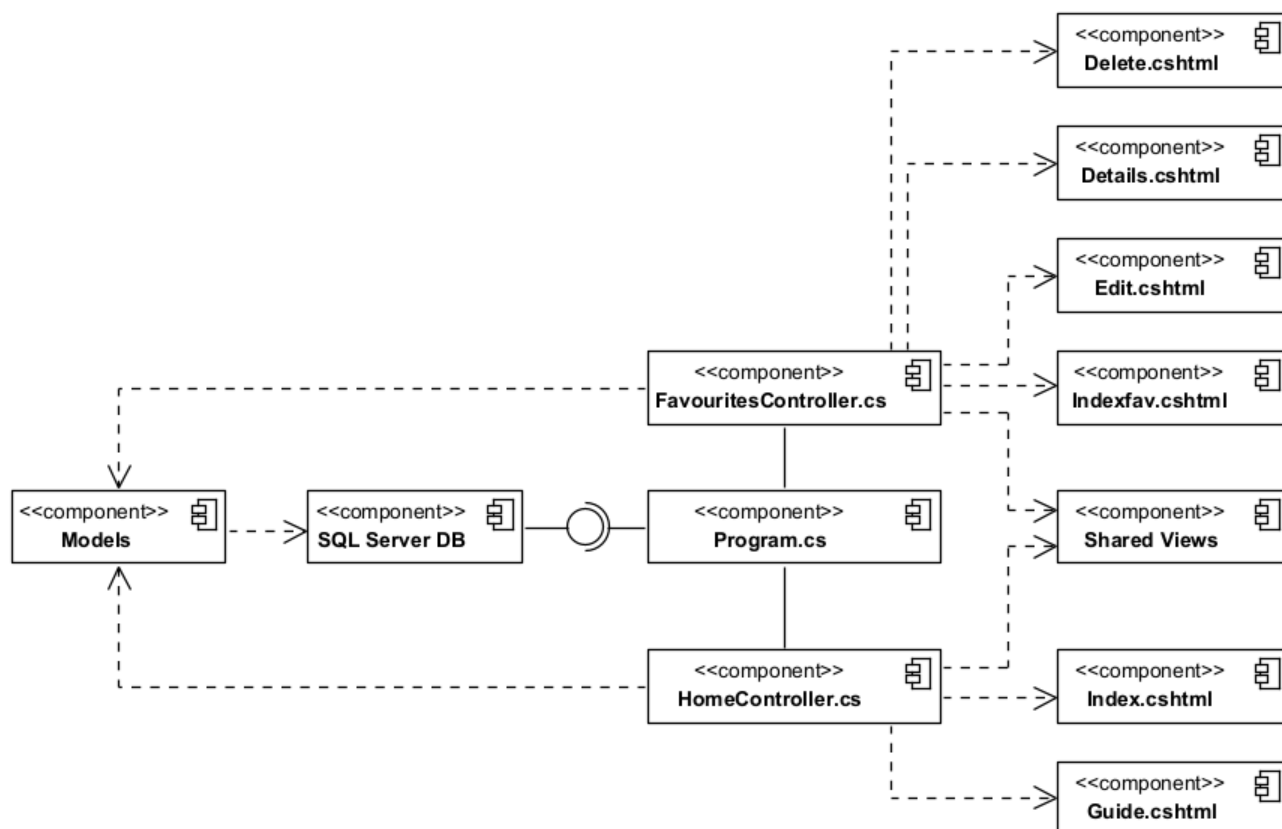


Рисунок 16 – Диаграмма компонентов приложения-агрегатора

Основная работа приложения происходит в контроллерах, особенностью разработки калькулятора ипотечного кредитования является алгоритм вычисления ипотечных параметров и подбора ипотечных предложений по введенным данным. Помимо контроллеров важную роль играют модели, отражающие структуру базы данных. Для поддержания актуальной структуры данных в соответствии с изменяющимися требованиями реализован механизм миграций, эти классы позволяют автоматизировать процесс обновления схемы данных, снижая риски потери информации. В качестве системы управления базами данных используется Microsoft SQL Server. В ней хранятся данные о банковских предложениях, а также информация о пользователе и выбранных им

предложениях. Код разработанного приложения-агрегатора для анализа условий и подбора ипотечных предложений приведен в приложении Б.

3.3 Реализация интерфейса приложения

Общий принцип интерфейса приложения состоит в переключении страниц на главном окне. Это позволяет вести независимую разработку каждой страницы отдельно, не нагружая при этом пространство логики для основного окна. Пользовательский интерфейс должен обеспечивать привычное и удобное представление информации, простое и эффективное выполнение основных функций пользователей. Разработка представлений проводилась с применением разметки Razor, в соответствии с выделенными ранее требованиями к пользовательскому интерфейсу.

В результате были реализованы страницы веб-приложения-агрегатора для анализа условий и подбора ипотечного кредитования:

- страница ипотечного калькулятора;
- страница справочной информации;
- страница регистрации;
- страница авторизации;
- страница «Избранного»;
- страница аккаунта пользователя.

Работа в приложении начинается с главной страницы ипотечного калькулятора (рисунок 17). На странице расположен непосредственно калькулятор ипотеки, пользователь вводит необходимые параметры, а именно стоимость жилья, размер первоначального взноса, желаемый срок кредита и процентную ставку. Кроме того, необходимо указать дату начала платежей, чтобы рассчитать график выплат, и выбрать тип ежемесячного платежа, по умолчанию выбран аннуитетный тип.

Общей чертой всех страниц приложения является панель меню, расположенная в верхней части экрана. Она предоставляет базовую навигацию по калькулятору.

The screenshot shows a mortgage calculator interface. At the top, there is a navigation bar with four items: '1 → Ipoteka' (with a right arrow), 'Справочник' (with a left arrow), '← 2' (with a left arrow), and '3 →' (with a right arrow). Below this is a descriptive text block: 'Расчет и подбор оптимального варианта, сравнение предложений разных банков. Доступен выбор 30 предложений в 20 банках с первоначальным взносом от 10 % и процентными ставками от 4,9% до 25,99%. Сделай первый шаг к своей мечте!'. The main form contains several input fields and sliders: '5 →' points to the 'Стоимость жилья:' field; '6 ←' points to the 'Первоначальный взнос: (0 %)' field; '7 →' points to the 'Срок кредита (лет):' field which has a slider set to 20; '8 ←' points to the 'Процентная ставка (%):' field which has a slider set to 16,5; '9 →' points to the 'Дата начала выплат:' field with a date format 'дд.мм.гггг'; '10 ↑' points to a calendar icon next to the date field; '11 ←' points to the 'Тип платежа:' dropdown menu which is set to 'Аннуитетный'; and '12 →' points to the 'Рассчитать' button.

Рисунок 17 – Страница ипотечного калькулятора

Основные элементы страницы ипотечного калькулятора:

- 1 – кнопка для перехода на главную страницу;
- 2 – кнопка для перехода на страницу справочной информации;
- 3 – кнопка для перехода на страницу регистрации;
- 4 – кнопка для перехода на страницу авторизации;
- 5 – поле ввода стоимости жилья;
- 6 – поле ввода первоначального взноса;
- 7 – поле ввода срока ипотечного кредита;
- 8 – поле ввода процентной ставки;
- 9 – поле ввода даты начала выплат;
- 10 – модуль календаря для выбора даты начала выплат;
- 11 – выпадающий список для выбора типа платежа;
- 12 – кнопка «Рассчитать».

Продолжением главной страницы является раздел «Часто задаваемые вопросы» (рисунок 18). Использование такого раздела помогает пользователям быстро освоиться в приложении и справиться с возможными трудностями при взаимодействии с незнакомой предметной областью. Список вопросов представлен в виде вертикально сложенного списка элементов, позволяет

сэкономить место на странице и сделать информацию более структурированной и читаемой.

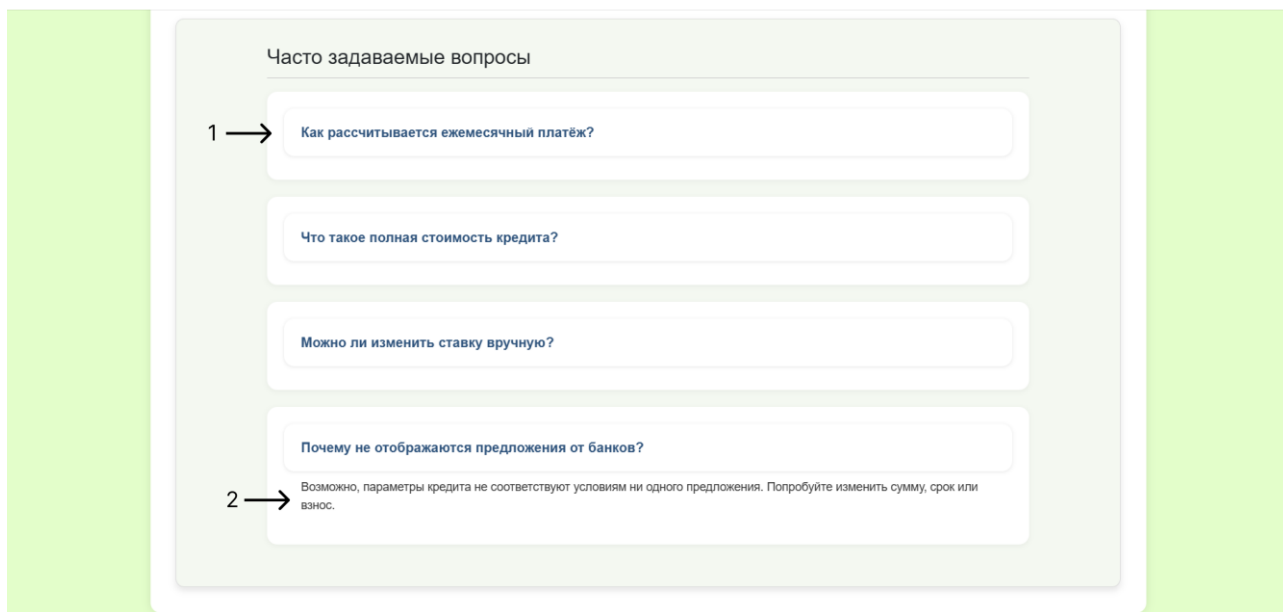


Рисунок 18 – Раздел «Часто задаваемые вопросы»

Основные элементы раздела «Часто задаваемые вопросы:

- 1 – кнопка для открытия вопроса;
- 2 – поле с ответом на выбранный вопрос.

Если пользователь не нашел ответ на возникший вопрос в данном разделе, он может перейти на страницу справочной информации для ознакомления с инструкцией и особенностями работы ипотечного калькулятора, интерфейс данной страницы представлен на рисунке 19.

В дальнейшем планируется заполнение данного раздела большим количеством информации, чтобы пользователь чувствовал себя более комфортно. Наличие достаточного объема справочной информации позволит сохранить количество пользователей, посетивших сайт, так как снимет необходимость в поиске незнакомой информации.

Как пользоваться ипотечным калькулятором

Этот калькулятор поможет вам рассчитать примерный ежемесячный платёж, переплату, необходимый доход и выбрать оптимальную ипотечную программу.

Пошаговая инструкция:

1. **Стоимость жилья** — введите стоимость объекта недвижимости.
2. **Первоначальный взнос** — укажите сумму или выберите процент от стоимости.
3. **Срок кредита** — укажите желаемый срок погашения ипотеки в годах.
4. **Процентная ставка** — используйте ползунок или ручной ввод для указания ставки.
5. **Дата начала выплат** — выберите дату, с которой начнётся погашение.
6. **Тип платежа** — выберите *аннуитетный* или *дифференцированный* платёж:

 **Аннуитетный** — ежемесячный платёж остаётся постоянным.

 **Дифференцированный** — платёж уменьшается с каждым месяцем.

После ввода данных нажмите **«Рассчитать»**, чтобы увидеть результаты:

 Ежемесячный платёж

 Переплата

 Необходимый доход

 График платежей

Ниже вы также увидите список актуальных ипотечных программ от банков, соответствующих введённым параметрам.

Если у вас есть вопросы, напишите нам по электронному адресу: sofiamixailovna00@mail.ru

Рисунок 19 – Страница справочной информации

Следующим элементов навигационной панели становится страница регистрации, представленная на рисунке 20.

Создайте аккаунт

Оформите регистрацию, чтобы сохранять избранные предложения и быстрее находить лучшие условия.

1 →

2 →

3 →

4 →

Рисунок 20 – Страница регистрации

Основные элементы страницы регистрации:

- 1 – поле для ввода электронной почты пользователя;
- 2 – поле для ввода пароля;
- 3 – поле для повторного ввода пароля;
- 4 – кнопка «Зарегистрироваться».

Для пользователей, которые уже имеют аккаунт в приложении-агрегаторе разработана страница авторизации (рисунок 21).

Вход в личный кабинет

Введите e-mail и пароль, указанные при регистрации. Или войдите через соцсеть.

Email

1 →

Пароль

2 →

3 → ☐ Запомнить меня

4 →

5 → [Нет аккаунта? Зарегистрируйтесь](#)

Рисунок 21 – Страница авторизации

Основные элементы страницы авторизации:

- 1 – поле для ввода электронной почты пользователя;
- 2 – поле для ввода пароля;
- 3 – кнопка для сохранения параметров входа;
- 4 – кнопка «Войти»;
- 5 – ссылка на страницу регистрации.

После успешного входа в аккаунт или регистрации пользователь возвращается на главную страницу ипотечного калькулятора, так же у него меняется состав панели навигации, кнопки входа и регистрации заменяются на кнопки перехода в раздел «Избранное», «Аккаунт пользователя» и кнопку «Выйти».

После указания необходимых данных в калькуляторе и нажатия кнопки «Рассчитать» как для авторизованных, так и для неавторизованных пользователей появятся результаты расчета в формате текста и круговой диаграммы, график платежей с возможностью скачивания, а так же список подобранных в соответствии с параметрами ипотечных предложений. На рисунке 22 представлен интерфейс главной страницы после введения данных и получения расчета.

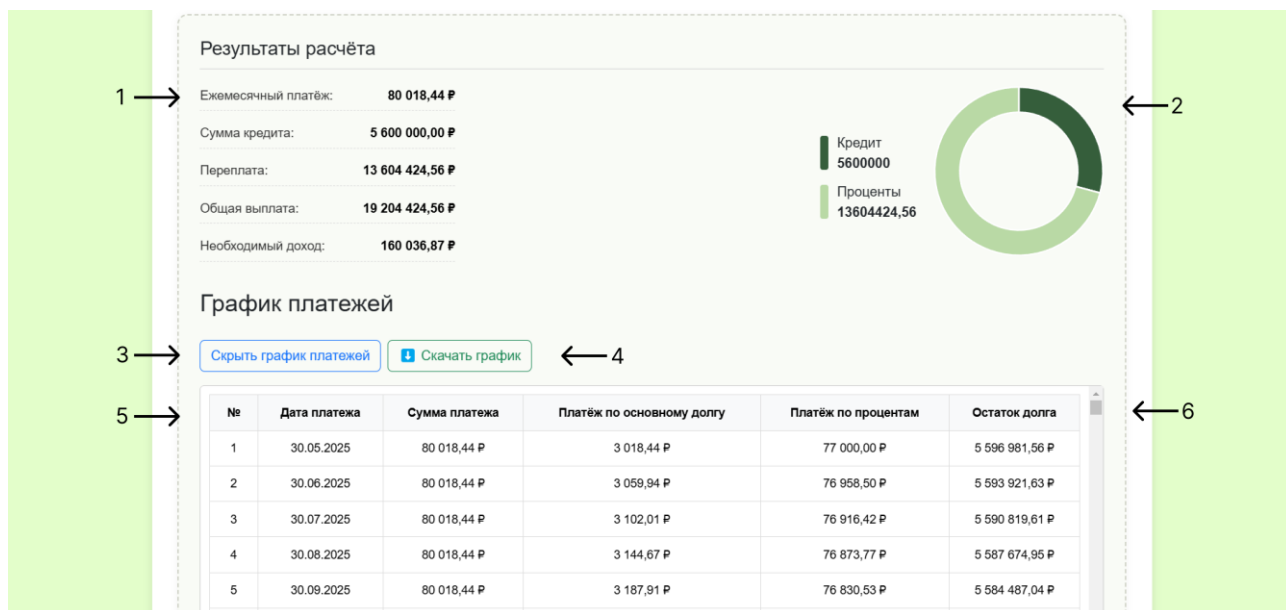


Рисунок 22 – Раздел «Результаты расчета»

Основные элементы раздела «Результаты расчета»:

- 1 – блок с результатами расчета ипотечных параметров;
- 2 – круговая диаграмма соотношения ипотеки к процентам по ипотеке;
- 3 – кнопка открытия/скрытия графика платежей;
- 4 – кнопка для скачивания графика платежей;
- 5 – таблица графика платежей;
- 6 – бегунок для навигации по таблице.

Ниже находится список ипотечных программ. Интерфейс страницы с подобранными программами представлен на рисунке 23. Раздел состоит из небольших блоков, каждый блок описывает то или иное банковское предложение. Пользователь видит название банка, название предложения, а так

же процентную ставку, полную стоимость кредита и ссылку для перехода на сайт банка.

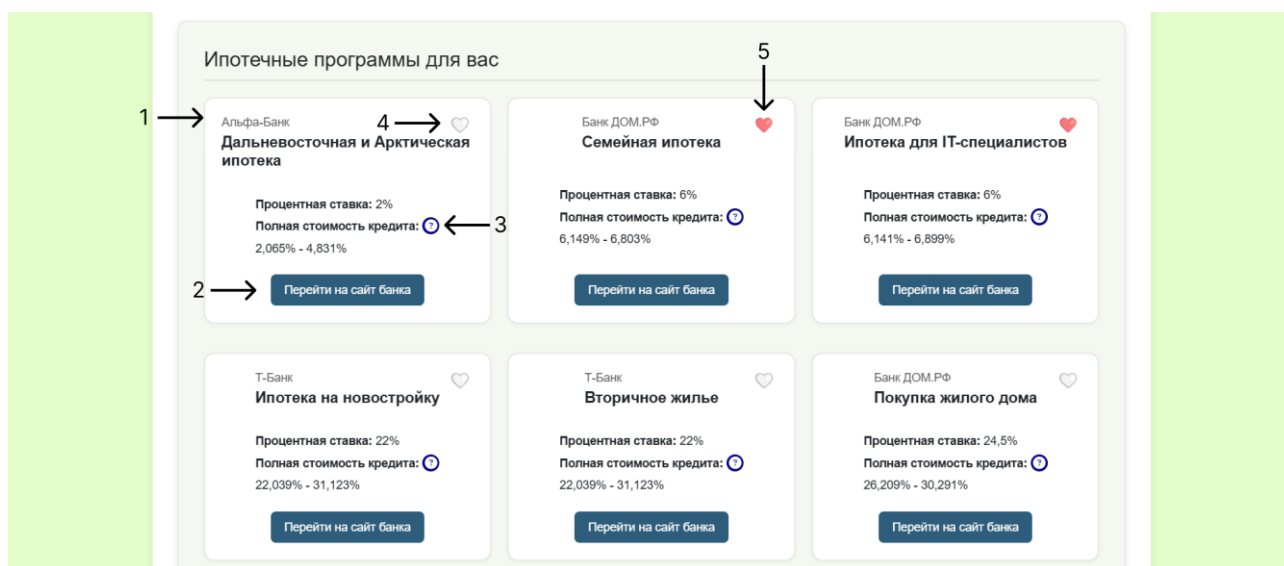


Рисунок 23 – Раздел «Ипотечные программы»

Основные элементы раздела «Результаты расчета»:

- 1 – блок с описанием банковского предложения;
- 2 – кнопка для перехода на сайт банка;
- 3 – всплывающее окно справочной информации;
- 4 – кнопка для сохранения предложения в «Избранное» в состоянии «не сохранено»;
- 5 – кнопка для сохранения предложения в «Избранное» в состоянии «сохранено».

В случае сохранения предложения неавторизованным пользователем появляется уведомление о необходимости войти или зарегистрироваться в системе для получения доступа к данному функционалу. Уведомления реализованы с помощью библиотеки SweetAlert2, которая позволяет создавать настраиваемые всплывающие окна в веб-приложениях.

После завершения работы с ипотечным калькулятором авторизованный пользователь может продолжить работу в системе в разделе «Избранное» (рисунок 24). На данной странице представлены все сохраненные пользователем

предложения. Для удобства реализована возможность добавления заметок, что упрощает работу по анализу и выбору конкретного предложения.

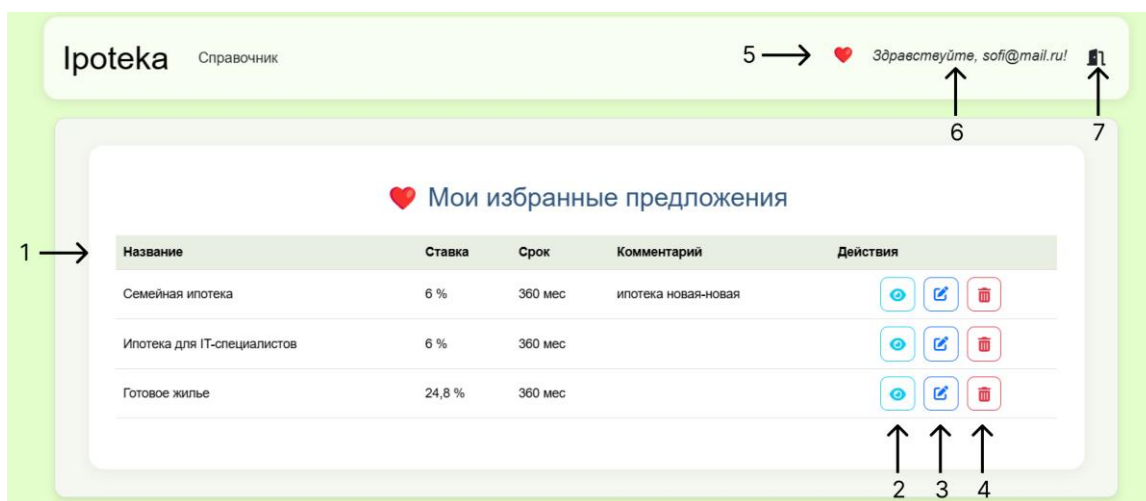


Рисунок 24 – Страница «Избранное»

Основные элементы страницы «Избранное»:

- 1 – таблица сохраненных предложений;
- 2 – кнопка для перехода на страницу «Детали предложения»;
- 3 – кнопка для перехода на страницу «Редактировать заметку»;
- 4 – кнопка для перехода на страницу «Удалить предложение»;
- 5 – кнопка для перехода в раздел «Избранное»;
- 6 – кнопка для перехода в «Аккаунт пользователя»;
- 7 – кнопка для выхода из аккаунта

Страница редактирования заметок предназначена улучшить условия использования системы и повысить эффективность пользователей. Интерфейс страницы представлен на рисунке 25.

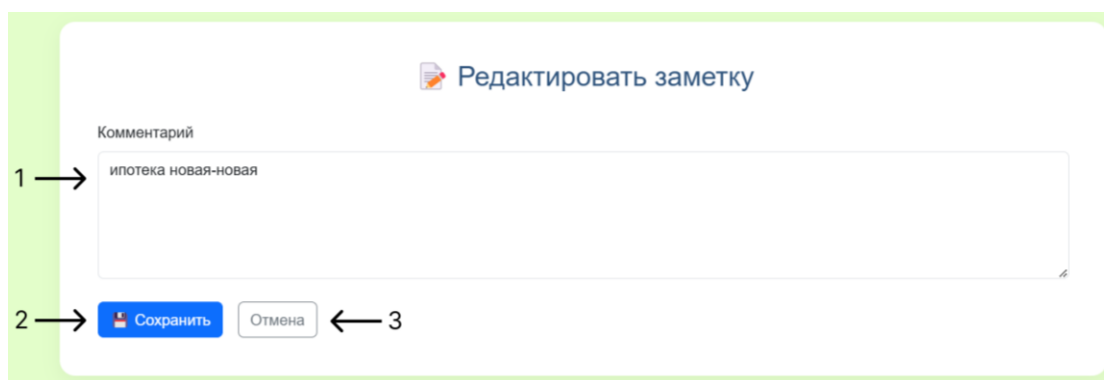


Рисунок 25 – Страница «Редактировать заметку»

Основные элементы страницы «Редактировать заметку»:

1 – поле для ввода текста заметки;

2 – кнопка «Сохранить»;

3 – кнопка «Отмена» изменений.

Страница «Детали предложения» содержит все имеющиеся сведения о банке и условиях ипотеки (рисунок 26).

Детали предложения: Семейная ипотека

Информация о банке
Банк: Банк ДОМ.РФ
Телефон: 88007758686
Адрес: 125009, г. Москва, ул. Воздвиженка, д. 10
Лицензия: 2312

Основные параметры
Ставка: 6,000 %
Стоимость жилья: от 6,149 до 6,803
Сумма кредита: от 500000 до 12000000
Первоначальный взнос: 20,000
Срок кредита: 36 – 360 лет
Стаж заемщика: 3 лет
Ссылка: <https://domrfbank.ru/mortgage/programs/family-mortgage/>

Тип платежа
Аннуитетный
размер ежемесячного платежа остается одинаковым в течение всего срока кредита (сначала платить проценты, а погашение основного долга оставить на более поздний срок)

Дополнительные условия
Актуальное
Предложение доступно и может быть выбрано клиентом
Начало действия: 25.06.2021 0:00:00
Окончание действия: 31.12.2030

Ваш комментарий
ипотека новая-новая

← Назад

Рисунок 26 – Страница «Детали предложения»

Если пользователь хочет удалить сохраненное банковское предложение, ему необходимо перейти на страницу удаления. Чтобы минимизировать вероятность случайного удаления предложения реализована отдельная страница,

содержащая основные параметры предложения, а также кнопки для удаления и отмены, интерфейс данной страницы представлен на рисунке 27.

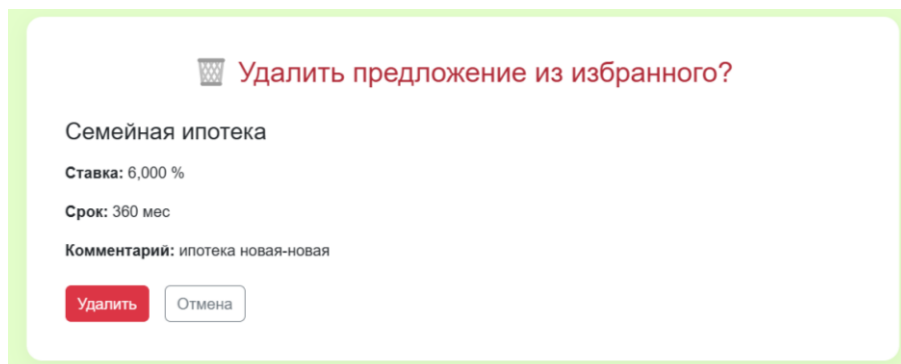


Рисунок 27 – Страница «Удалить предложение»

Авторизованным пользователем доступен функционал управления аккаунтом, в разделе «Профиль» реализована возможность добавления и изменения номера телефона (рисунок 28).

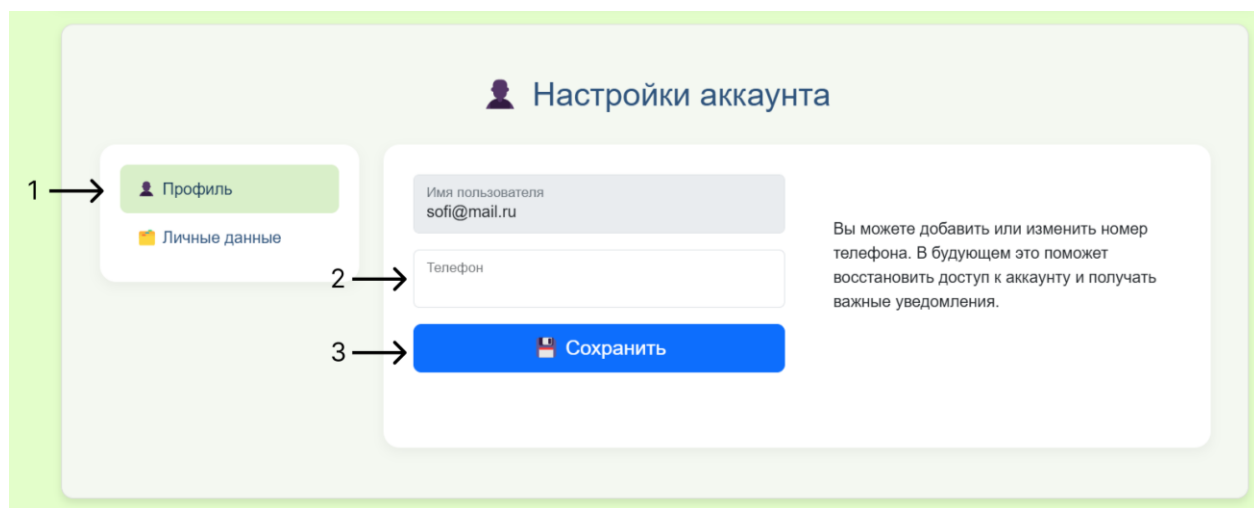


Рисунок 28 – Раздел «Профиль»

Основные элементы страницы «Настройки аккаунта»:

- 1 – навигация по аккаунту пользователя;
- 2 – поле для ввода номера телефона;
- 3 – кнопка «Сохранить».

При переходе в раздел «Личные данные» становятся доступны возможности скачать информации об аккаунте и удалить аккаунт (рисунок 29).

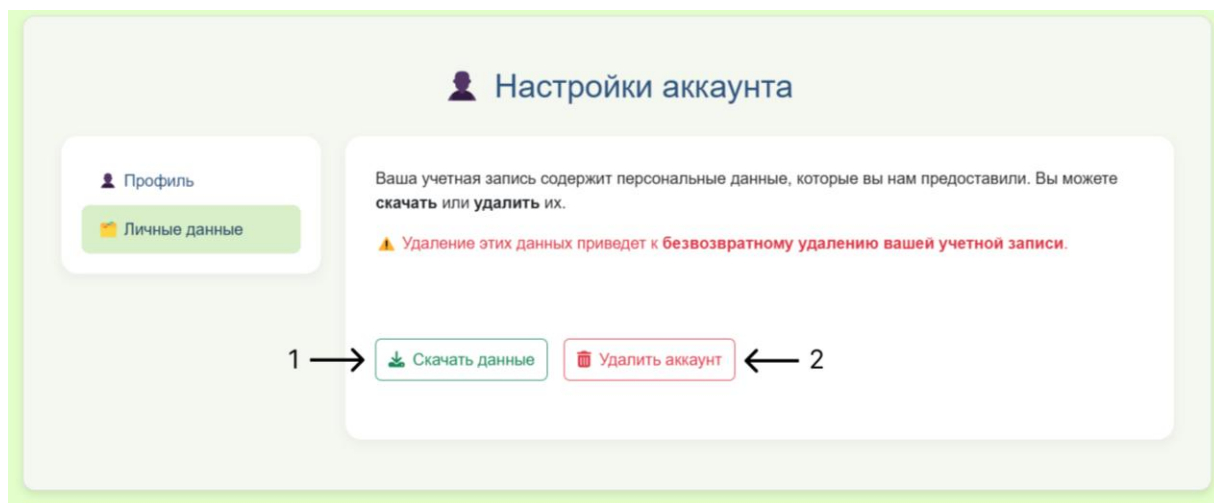


Рисунок 29 – Раздел «Личные данные»

Основные элементы страницы «Настройки аккаунта»:

1 – кнопка для скачивания данных об аккаунте;

2 – кнопка для удаления аккаунта.

Разработка графического интерфейса производилась с использованием каскадной таблицы стилей CSS3 и разметки Razor. Razor служит для формирования структуры веб-страниц. CSS3, в свою очередь, применяется для настройки дизайна, включая параметры шрифтов, цветовую гамму и многое другое.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования и метрик кода

Тестирование – важная часть процесса разработки программного обеспечения, оно проводится на разных этапах проекта. Тестирование программного обеспечения – это метод оценки функциональности программного обеспечения. В ходе этого процесса проверяется, соответствует ли фактическое программное обеспечение ожидаемым требованиям, и обеспечивается отсутствие ошибок в программном обеспечении. Целью тестирования является выявление ошибок, сбоев или недостающих требований по сравнению с фактическими требованиями. В основном оно направлено на измерение спецификации, функциональности и производительности программного обеспечения или приложения [16].

Наиболее распространены такие виды тестирования, как функциональное, модульное, интеграционное и нагрузочное.

Функциональное тестирование проверяет систему на соответствие функциональным требованиям и спецификациям, гарантирует их должное удовлетворение, фокусируется на моделировании реального использования системы. Модульное тестирование направлено на проверку отдельных небольших частей приложения, которые можно исследовать изолированно от других подобных частей [17].

Интеграционное тестирование – это метод проверки взаимодействия различных модулей или компонентов программного приложения друг с другом. Нагрузочное тестирование – это вид тестирования производительности, который имитирует реальную нагрузку на систему или приложение, чтобы проверить, как они работают в условиях стресса [18].

Для полного охвата взаимодействия пользователя с приложением необходимо написать тест-кейсы с детальным описанием действий для проверки конкретных функций, основанные на пользовательских сценариях.

Главным основанием для выбора функционального тестирования в качестве метода ручного тестирования послужило большое внимание требований к программному продукту, обращенных на функциональные аспекты. В качестве автоматизированного тестирования было выбрано модульное тестирование, юнит тесты. Данный метод позволяет эффективно и в короткий срок провести проверку отдельных компонентов системы.

Возросшая сложность современных программных приложений также усложняет задачу обеспечения надёжности и удобства сопровождения кода. Метрики кода – это набор программных показателей, которые дают лучшее представление о коде, позволяют понять, какие типы или методы следует переработать или протестировать более тщательно [19].

Для оценки качества кода разработанного приложения-агрегатора для анализа условий и подбора ипотечных предложений были выбраны такие метрики, как цикломатическая сложность, метрика Холстеда, а также метрика количества строк кода (SLOC).

Цикломатическая сложность – мера структурной сложности кода. Она рассчитывается путем подсчета количества различных путей кода в потоке программы. Программные метрики Холстеда – это набор показателей, основанных на количестве различных операторов и операндов в программе, используется для оценки усилий, необходимых для разработки и поддержки программы. Метрика SLOC используется для количественной оценки размера и сложности программного обеспечения. Она измеряет количество строк в исходном коде программы, включая комментарии и пустые строки [20, 21].

4.2 Тестирование приложения-агрегатора для анализа условий и подбора ипотечных предложений

4.2.1 Тестирование пользовательской части приложения для анализа условий и подбора ипотечных предложений

Для проведения функционального тестирования пользовательской части был выбран метод тестирования черным ящиком. Пользовательская часть

приложения в рамках основного функционала представлена вводом стоимости жилья, первоначального взноса, срока, ставки, даты начала выплат. Для ввода стоимости жилья имеются определенные требования, которым соответствуют выделенные классы эквивалентности в таблице 12.

Таблица 12 – Классы эквивалентности

Требование	Правильные КЭ	Неправильные КЭ
Длина адреса	7–∞ (1)	0–6 (2)
Содержание	Цифры (3)	Остальные (4)
Структура	Первая цифра от 1 до 9 (5)	Первая цифра 0 (6)

Согласно выделенным классам эквивалентности были составлены тест-кейсы, приведенные в таблице 13. Результаты приведены в приложении Г.

Таблица 13 – Тест-кейсы для выделенных классов эквивалентности

№	КЭ	Входные данные	Выходные данные
1	1,3,5	5600000	Успешно
2	2,3,5	1000	Ошибка: Длина меньше 7 символов (рисунок Г.1)
3	1,4,5	1миллион	Пользователю недоступен ввод букв (рисунок Г.2)
4	1,3,6	05230780	Ошибка: Первая цифра 0, игнорируется расчет произведен по оставшимся цифрам (рисунок Г.3)
5	2,4,6	01мп-	Ошибка: Длина меньше 7, содержит не только цифры, начинается с нуля. Буквы автоматически игнорируются (рисунок Г.4)
6	4,6	0ждлд=-09876кампгрл	Ошибка: Длина больше 9, содержит не только цифры, начинается с нуля (рисунок Г.5)

В дополнение к выделенным класса эквивалентности были составлены граничные условия к вводимым данным. Граничные условия к вводу стоимости жилья представлены в таблице 14.

Таблица 14 – Граничные условия

Границы	Содержание
Физические	Строка неотрицательной длины (Ф)
Логические	Сумма не меньше нуля (Л1) Сумма не равна нулю (Л2) Сумма, не начинающаяся с нуля (Л3) Не указан город (Л4) Не указана улица (Л5) Номер дома не является целым положительным числом (Л6)
Технологические	Сумма является не целым числом (Т1) Целое число не больше максимально возможного по типу данных (Т2)

Для выделенных граничных условий были составлены дополнительные тест-кейсы, приведенные в таблице 15.

Таблица 15 – Тест-кейсы для выделенных граничных условий

№	Границы	Входные данные	Выходные данные
1	Ф1, Л1, Л2, Л3, Т1	8500000	Успешно
2	Ф1	{ }	Ошибка: Отсутствуют символы (рисунок Г.6)
3	Л1	-1000000	Ошибка: Указано отрицательное число (рисунок Г.7)
4	Л2, Л3	0	Ошибка: Число равно нулю и начинается с нуля (рисунок Г.8)

Продолжение таблицы 15

№	Границы	Входные данные	Выходные данные
5	T1	4312000,58	Ошибка: Число не является целым (рисунок Г.9)
6	T2	893248796201599785200598	Ошибка: Размер числа больше допустимого по типу (рисунок Г.10)

Полученные результаты соответствуют ожидаемым, а значит, пользовательская часть приложения работает корректно.

4.2.2 Тестирование серверной части приложения для анализа условий и подбора ипотечных предложений

Для обеспечения корректности работа разработанного программного продукта и проверки ожидаемого поведения программы проведено модульное тестирование. Тестирование компонентов расчета ипотечных параметров и подбора банковских предложений на основании введенных параметров осуществляется с помощью фреймворка xUnit. Полный код модульных тестов приведен в приложении Д.

План модульного тестирования представлен в таблице 16.

Таблица 16 – План тестирования

Тестируемый компонент	Тест	Ожидаемый результат
MortgageCalculator	Должен создаваться	Пройден
	Должен корректно рассчитывать сумму кредита	Пройден
	Должен корректно рассчитывать ежемесячный платеж	Пройден
	Должен корректно рассчитывать общую сумму выплат	Пройден
	Должен корректно рассчитывать необходимый доход	Пройден

Продолжение таблицы 16

Тестируемый компонент	Тест	Ожидаемый результат
BankOffers	Должен возвращать список банковских предложений	Пройден
	Должен обрабатывать ситуации, когда нет подходящих предложений	Пройден
Validation	Должен корректно обрабатывать отрицательные или некорректные данные	Пройден
	Должен корректно обрабатывать нулевые значения	Пройден

Результат выполнения тест-кейсов представлен на рисунке 30.

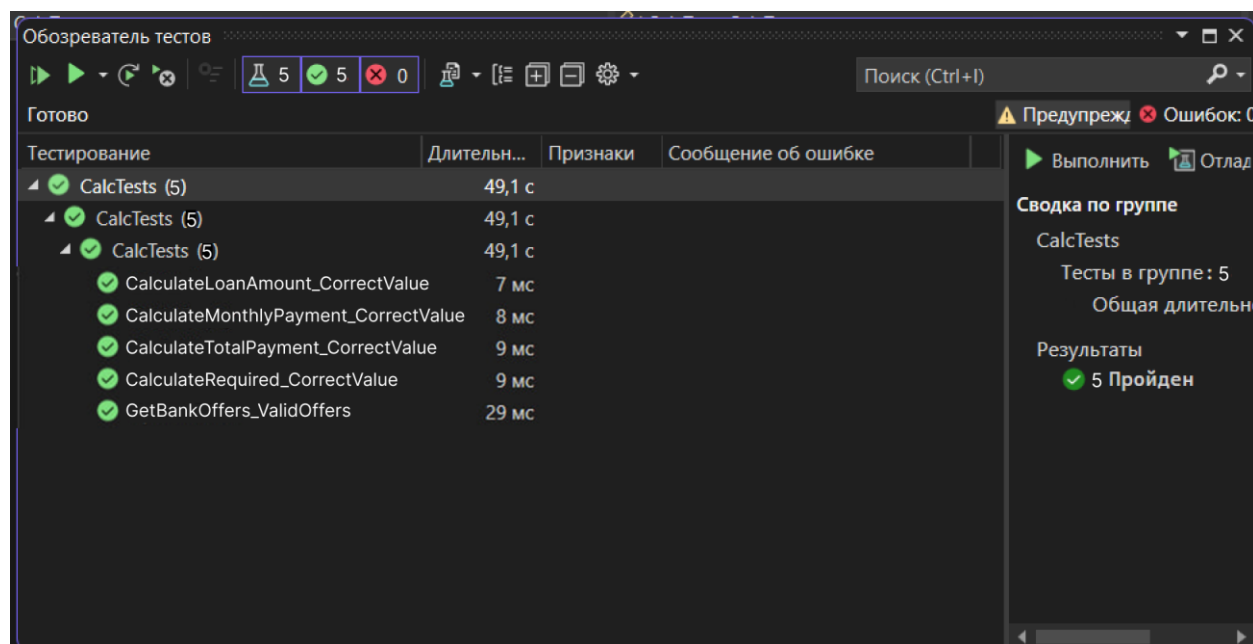


Рисунок 30 – Результат модульного тестирования

Программный продукт ипотечный калькулятор успешно прошел все этапы модульного тестирования.

4.3 Оценка качества программного кода

4.3.1 Цикломатическая сложность

Цикломатическая сложность является одной из метрик программного

обеспечения, которая позволяет количественно оценить логическую сложность программы. Для вычисления цикломатической сложности на примере функции расчета ипотечных условий был использован граф потока управления программы, представленный на рисунке 31.

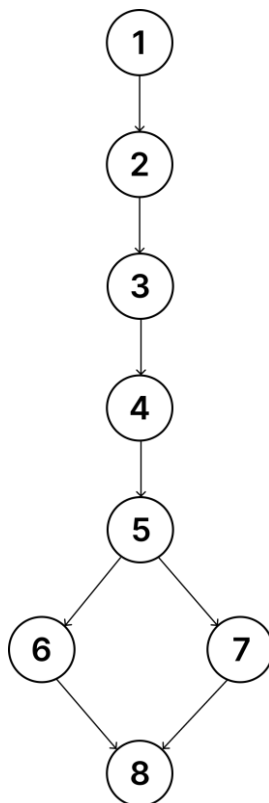


Рисунок 31 – Управляющий граф прецедента «Расчет ипотечных параметров»

Для вычисления цикломатической сложности $Z(G)$ применяется следующая формула:

$$Z(G) = e - v + 2p,$$

где e – число дуг ориентированного графа G ;

v – число вершин;

p – число компонентов связности графа.

Функциональный граф, построенный для алгоритма расчета ипотечных параметров, состоит из 8 дуг и 8 вершин. Для простой программы, или подпрограммы, или метода число компонентов связности графа всегда равно 1. На основании этих данных была определена циклическая сложность метода.

Ее значения получилось равным 2.

Данный результат позволяет говорить, что метод обладает средней структурной сложностью.

4.3.2 Метрика Холстеда

Для расчета метрик Холстеда рассматривается функция Calculate компонента HomeController. Функция реализует расчет основных ипотечных параметров, а также подбор банковских предложений, соответствующих введенным данным пользователя. Код данной функции представлен в листинге 1.

Листинг 1 – Исходный код функции Calculate

```
[HttpPost]
public async Task<IActionResult> Calculate(HomeLoan model)
{
    decimal loanAmount = model.HomePrice - model.Payment;
    decimal monthlyInterestRate = model.Rate / 1200;
    int term = model.Term;
    model.MonthlyPayment = loanAmount * (monthlyInterestRate *
(decimal)Math.Pow((double)(1 + monthlyInterestRate), term)) /
((decimal)Math.Pow((double)(1 + monthlyInterestRate), term) - 1);
    Model.TotalPayment = model.MonthlyPayment * term;
    Model.Overpayment = model.TotalPayment - loanAmount;
    model.LoanAmount = loanAmount;

    model.BankOffers = await _context.Offer
        .Where(o => o.InterestRate <= model.Rate &&
o.LoanAmountMax >= loanAmount)
        .ToListAsync();
    Return View(model);
}
```

Для вычисления данной метрики требуется посчитать число вхождения уникальных операторов (N1) и операндов (N2), а также общее количество операторов (n1) и операндов (n2).

Операторы функции Calculate и число их вхождений приведены в таблице 17.

Таблица 17 – Операторы функции Calculate

№	Операторы	Число вхождений
1	ToListAsync	1
2	Where	1
3	Math	2
4	Pow	2
5	View	1
6	_context	1
7	await	1
8	return	1
9	.	19
10	=	8
11	*	3
12	/	2
13	-	3
14	+	2
15	=>	1
16	<=	1
17	>=	1
18	{	1
19	}	1
20	(	14
21	)	14
22	;	9
23	,	2
	n1 = 23	N1 = 91

Операнды функции Calculate и число их вхождений приведены в таблице 18.

Таблица 18 – Операнды функции Calculate

№	Операнды	Число вхождений
1	model	13
2	loanAmount	5
3	monthlyInterestRate	4
4	term	4
5	HomePrice	1

Продолжение таблицы 18

№	Операнды	Число вхождений
6	Payment	1
7	Rate	2
8	Term	1
9	MonthlyPayment	2
10	TotalPayment	2
11	Overpayment	1
12	LoanAmount	1
13	BankOffers	1
14	Offer	1
15	InterestRate	1
16	LoanAmountMax	1
17	o	3
18	1200	1
19	1	3
	n1 = 19	N1 = 48

Данные, полученные в ходе обработки программного кода функции Calculate компонента HomeController были использованы для расчета характеристик программы по формулам из метрики Холстеда. Результаты вычислений представлены в таблице 19.

Таблица 19 – Результаты вычислений по метрике Холстеда

Значение	Результат
Длина реализации программы	139
Словарь программы	42
Вычислительная сложность	749.21
Структурная сложность	1
Количество строк кода	17
Комплексный показатель качество	90.46

Все полученные показатели программы находятся в пределах нормы и удовлетворяют требованиям по метрикам программного кода. Комплексный показатель кода позволяет определить относительную сложность поддержки кода и составляет 90.46. Полученное значения находится в допустимых границах и означает низкую сложность поддержки кода. Таким образом, алгоритм кода имеет приемлемый индекс сопровождаемости.

4.3.3 Метрика SLOC

Расчет метрики SLOC будет производиться над компонентом, отвечающим за расчет ипотечных параметров, исходный код данного модуля представлен в листинге 1.

В данной программе имеем:

- 17 физических строк кода;
- 10 логических строк кода;
- 0 строк комментариев.

Данный показатель говорит о низком объеме кода, из чего можно сделать вывод о среднем уровне трудозатрат для реализации алгоритма.

Заключение

В рамках выпускной квалификационной работы спроектировано и разработано приложение-агрегатор для анализа условий и подбора ипотечных предложений.

На стадии анализа требований проведен анализ предметной области, сравнительный анализ приложений аналогичного назначения, а также выявлены функциональные и нефункциональные требования для разработки нового программного обеспечения. Кроме того, изучены особенности расчета ипотечных кредитных обязательств с помощью калькуляторов ипотечного кредитования.

Произведено планирование и оценка стоимости разработки приложения-агрегатора. Затраты на проект составили 98630 рублей. Начало работ было определено как 8 апреля, а окончание работ по проекту запланировано на 3 июня текущего года. Затраты трудовых ресурсов составили 392 часа.

На стадии проектирования выполнено разработка моделей базы данных, архитектуры программного обеспечения, выделены классы разрабатываемого приложения и сформулированы требования к пользовательскому интерфейсу.

Разработка приложения-агрегатора для анализа условий и подбора ипотечных предложений включала в себя реализацию базы данных в SQL Server, а также реализацию основного приложения с использованием ASP.NET Core MVC в среде разработки Visual Studio 2022.

Контроль качества разработанного веб-приложения был проведен с помощью функционального и модульного методов тестирования. Кроме того, выполнен анализ метрик кода: цикломатической сложности, метрик Холстеда и SLOC. В результате тестирования выявлена корректность работы программы и ее соответствие функциональным требованиям, подсчет метрик показал достаточный уровень разработанного приложения.

Таким образом, в результате выполнения выпускной квалификационной работы разработано приложение-агрегатор для анализа условий и подбора

ипотечных предложений, предоставляющее удобный способ расчета приблизительной величины основных ипотечных параметров и подбора банковских предложений без необходимости производить сложные математические вычисления и анализировать большое количество информации.

Список использованных источников

1. Голосова, С.М Разработка приложения-агрегатора подбора ипотеки / С.М Голосова, И.Ю. Балашова // Новые информационные технологии и системы: материалы XXI международной науч.-техн. конф. – Пенза: Изд-во Вестник Пензенского государственного университета. 2025. № 2. С. 44–47.
2. Голосова, С.М. Разработка ипотечного калькулятора / Голосова С.М., Балашова И.Ю. // Информационные технологии в науке и образовании. Проблемы и перспективы: материалы XI всероссийской науч.-практ. конф. – Пенза: Изд-во Пензенского государственного университета, 2025.
3. Диаграмма вариантов использования – Унифицированный язык моделирования (UML) // Обучающий портал «GeeksforGeeks». [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/use-case-diagram/> (дата обращения 04.05.2025 г.).
4. Балашова, И. Ю. Современные информационные технологии в проектировании программных систем и комплексов : учеб. пособие / И. Ю. Балашова, Д. В. Такташкин; под ред. П. П. Макарычева. – Пенза : Изд-во ПГУ, 2019. – 106 с.
5. Монолитная архитектура – Системное Проектирование. // Обучающий портал «GeeksforGeeks». [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/monolithic-architecture-system-design/> (дата обращения: 28.12.2024).
6. Введение в ASP.NET Базовый фреймворк MVC. // DotNetTutorials. [Электронный ресурс]. URL: <https://dotnettutorials.net/lesson/introduction-asp-net-core-mvc/> (дата обращения: 28.12.2024).
7. Дамиаре Кешинро, «Что такое концептуальная модель данных и почему она важна при разработке SQL?». // Baeldung [Электронный ресурс]. URL: <https://www.baeldung.com/sql/design-conceptual-data-model> (дата обращения: 22.05.2025).

8. Моделирование данных: обзор // Тематический блог «Хабр». [Электронный ресурс]. URL: <https://habr.com/ru/articles/556790/> (дата обращения: 19.10.2024).
9. Диаграмма классов. Унифицированный язык моделирования (UML). // Обучающий портал «GeeksforGeeks». [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/> (дата обращения: 24.05.2025).
10. Процесс разработки пользовательского интерфейса: 6 простых шагов для создания отличного пользовательского интерфейса. // Обучающий портал «GeeksforGeeks». [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/ui-design-process-everything-you-need-to-know/> (дата обращения: 25.05.2025).
11. Дино Эспозито, «Обзор механизма Razor View в ASP.NET MVC» – Статья // Redgate [Электронный ресурс]. URL: <https://www.red-gate.com/simple-talk/development/dotnet-development/a-look-at-the-razor-view-engine-in-asp-net-mvc/> (дата обращения: 25.05.2025).
12. Часть 3. Добавление представления в приложение MVC ASP.NET Core // Официальная документация Microsoft. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/first-mvc-app/adding-view?view=aspnetcore-9.0&tabs=visual-studio> (дата обращения: 25.05.2025).
13. Глава 14. Проектирование и разработка баз данных. Руководство для разработчиков баз данных. Концептуальная, логическая и физическая модели данных. // Официальная документация Visual Paradigm. [Электронный ресурс]. URL: <https://circle.visual-paradigm.com/docs/database-design-engineering/database-designers-guide/conceptual-logical-and-physical-data-model/> (дата обращения: 27.05.2025).
14. Миграция и заполнение данных – (ASP.NET MVC Entity Framework). // Платформа «Additional Sheet» [Электронный ресурс]. URL: <https://additionalsheet.com/aspnet-mvc-migrations-and-seeding-data> (дата обращения: 27.05.2025).

15. Разработка приложений MVC ASP.NET Core // Официальная документация Microsoft [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/develop-asp-net-core-mvc-apps> (дата обращения: 28.05.2025).

16. Учебное пособие по тестированию программного обеспечения. // Обучающий портал «GeeksforGeeks». [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/software-testing-tutorial/> (дата обращения: 26.05.2025).

17. Куликов С. Тестирование программного обеспечения. Базовый курс (1- изд.). – Минск.: EPAM Systems, 2016. – 300 с.

18. Виды тестирования программного обеспечения. // Обучающий портал «GeeksforGeeks». [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/types-software-testing/> (дата обращения: 26.05.2025).

19. Значения метрик кода // Официальная документация Microsoft [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-values?view=vs-2022> (дата обращения: 26.05.2025).

20. Метрики программного обеспечения Холстеда – разработка программного обеспечения. // Обучающий портал «GeeksforGeeks». [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/> (дата обращения: 26.05.2025).

21. Строки кода SLOC: раскрываем скрытые показатели качества. // Metridev. [Электронный ресурс]. URL: <https://www.metridev.com/metrics/sloc-lines-of-code-unveiling-the-hidden-quality-metrics/> (дата обращения: 26.05.2025).

Приложение А
(справочное)
Глоссарий

БД (База данных) – это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

СУБД (Система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Microsoft SQL Server (MS SQL Server) – это реляционная система управления базами данных (СУБД) от компании Microsoft, предназначена для хранения, обработки и анализа информации в различных сферах – от веб-приложений до корпоративных информационных систем.

Фреймворк – программная платформа, задающая архитектурные принципы и общие правила разработки приложений в определенной предметной области.

Веб-браузер – прикладное программное обеспечение для поиска и просмотра веб-страниц, а также управления веб-приложениями;

Веб-приложение – это клиент-серверное программное решение, доступ к которому осуществляется через сетевой браузер посредством передачи данных по протоколу HTTP, размещается на удаленном сервере

AJAX (Asynchronous JavaScript and XML) – это веб-технология, которая позволяет обмениваться запросами с сервером, отправлять и получать от него данные без перезагрузки страницы.

.NET – это платформа для разработчиков, состоящая из инструментов, языков программирования и библиотек для создания множества различных типов приложений. ASP.NET Core расширяет платформу разработчика .NET инструментами и библиотеками, специально предназначенными для создания веб-приложений.

MVC – это архитектурный паттерн, предписывающий разделение программного кода на три взаимосвязанных компонента: модель данных, представление для визуализации информации и контроллер для обработки пользовательских запросов.

Приложение Б
(обязательное)
Листинг приложения

Листинг Б.1 – Файл HomeController.cs

```

using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;
using MortgageCalc.Models;
using MortgageCalc.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Identity;

namespace MortgageCalc.Controllers
{
    public class HomeController : Controller
    {
        private readonly ApplicationDbContext _context;

        private readonly UserManager<IdentityUser> _userManager;
        public HomeController(ApplicationDbContext context,
            UserManager<IdentityUser> userManager)
        {
            _context = context;

            _userManager = userManager;

            [HttpGet]
            public IActionResult Index()
            {
                return View();
            }

            [HttpPost]
            public async Task<IActionResult> Index(decimal
propertyPrice, decimal initialPayment, int loanTerm, decimal
interestRate, DateTime startDate, string paymentType)
            {
                var creditAmount = propertyPrice - initialPayment;
                var months = loanTerm * 12;
                var monthlyRate = 16.5m;
                if (interestRate != 0)
                {
                    monthlyRate = (interestRate / 100) / 12;
                }
                else
                {
                    monthlyRate = (monthlyRate / 100) / 12;
                }

                decimal monthlyPayment = 0;
                if (paymentType == "annuity")
                {

```

```

        monthlyPayment = creditAmount * (monthlyRate *
(decimal)Math.Pow((double)(1 + monthlyRate), months)) /
        ((decimal)Math.Pow((double)(1 +
monthlyRate), months) - 1);
    }
    else if (paymentType == "differentiated")
    {
        decimal principalPart = creditAmount / months;
        decimal remainingCredit = creditAmount;
        List<decimal> monthlyPayments = new();

        for (int month = 1; month <= months; month++)
        {
            decimal interestPart = remainingCredit *
monthlyRate;
            decimal monthly = principalPart +
interestPart;
            monthlyPayments.Add(monthly);

            remainingCredit -= principalPart;
        }

        monthlyPayment = monthlyPayments.First();
    }

    decimal totalPayment = monthlyPayment * months;
    decimal overpayment = totalPayment - creditAmount;
    decimal requiredIncome = monthlyPayment * 2;

    var schedule = new List<PaymentScheduleItem>();
    decimal remainingDebt = creditAmount;

    for (int i = 0; i < months; i++)
    {
        DateTime paymentDate = startDate.AddMonths(i);
        decimal interestPayment = remainingDebt *
monthlyRate;
        decimal principalPayment = 0;

        if (paymentType == "annuity")
        {
            principalPayment = monthlyPayment -
interestPayment;
        }
        else
        {
            principalPayment = creditAmount / months;
            monthlyPayment = principalPayment +
interestPayment;
        }

        remainingDebt -= principalPayment;
    }

```

```

        schedule.Add(new PaymentScheduleItem
        {
            PaymentDate = paymentDate,
            PaymentAmount = Math.Round(monthlyPayment, 2),
            PrincipalPayment =
Math.Round(principalPayment, 2),
            InterestPayment = Math.Round(interestPayment,
2),
            RemainingDebt =
Math.Max(Math.Round(remainingDebt, 2), 0)
        });
    }

    decimal downPaymentPercent = (initialPayment /
propertyPrice) * 100;
    var offersQuery = _context.DbOffer
.Include(o => o.Bank)
.Where(o =>
    creditAmount >= o.LoanAmountMin &&
    creditAmount <= o.LoanAmountMax &&
    months >= o.TermMin &&
    months <= o.TermMax &&
    downPaymentPercent >= o.DownPayment);
    var offers = await offersQuery
.Select(o => new OfferWithBankDto
{
    Id = o.Id,
    Title = o.Title,
    InterestRate = o.InterestRate,
    DownPayment = o.DownPayment,
    LoanCostMin = o.LoanCostMin,
    LoanCostMax = o.LoanCostMax,
    Link = o.Link,
    BankTitle = o.Bank.Title
})
.OrderBy(o => o.InterestRate)
.ToListAsync();

    if (!offers.Any())
    {
        ViewBag.NoOffersMessage = "К сожалению, не найдено
подходящих ипотечных предложений.";
    }
    var model = new MortgageCalculatorViewModel
    {
        MonthlyPayment = Math.Round(monthlyPayment, 2),
        CreditAmount = creditAmount,
        Overpayment = Math.Round(overpayment, 2),
        TotalPayment = Math.Round(totalPayment, 2),
        RequiredIncome = Math.Round(requiredIncome, 2),
        PaymentSchedule = schedule,
        Offers = offers,
    }

```

```

        FavouriteOfferIds = new List<int>()
    };
    if (User.Identity.IsAuthenticated)
    {
        var userId = _userManager.GetUserId(User);
        model.FavouriteOfferIds = await
_context.DbFavourite
        .Where(f => f.UserId == userId)
        .Select(f => f.OfferId)
        .ToListAsync();
    }
    return View("Index", model);
}

public IActionResult Guide()
{
    return View();
}

[ResponseCache(Duration = 0, Location =
ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
    return View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}
}

```

Листинг Б.2 – Файл FavouritesController.cs

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MortgageCalc.Data;
using MortgageCalc.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;

namespace MortgageCalc.Controllers
{
    [Authorize]
    public class FavouritesController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly UserManager<IdentityUser> _userManager;

        public FavouritesController(ApplicationDbContext context,
UserManager<IdentityUser> userManager)
        {
            _context = context;
            _userManager = userManager;
        }
    }
}

```

```

[HttpPost]
public async Task<IActionResult> Add(int offerId)
{
    var user = await _userManager.GetUserAsync(User);
    var alreadyAdded = _context.DbFavourite.Any(f =>
f.UserId == user.Id && f.OfferId == offerId);
    if (!alreadyAdded)
    {
        var favorite = new Favourite
        {
            UserId = user.Id,
            OfferId = offerId
        };
        _context.DbFavourite.Add(favorite);
        await _context.SaveChangesAsync();
    }

    return RedirectToAction("Index", "Home");
}

public async Task<IActionResult> Index()
{
    var userId = _userManager.GetUserId(User);
    var favourites = await _context.DbFavourite
        .Include(f => f.Offer)
        .Where(f => f.UserId == userId)
        .ToListAsync();

    return View(favourites);
}

public async Task<IActionResult> Details(int? id)
{
    if (id == null) return NotFound();

    var userId = _userManager.GetUserId(User);
    var favourite = await _context.DbFavourite
        .Include(f => f.Offer)
        .ThenInclude(o => o.Bank)
        .Include(f => f.Offer)
        .ThenInclude(o => o.PaymentType)
        .Include(f => f.Offer)
        .ThenInclude(o => o.DataCondition)
        .ThenInclude(dc => dc.Condition)
        .FirstOrDefaultAsync(m => m.Id == id && m.UserId
== userId);
    if (favourite == null) return NotFound();

    return View(favourite);
}

public async Task<IActionResult> Edit(int? id)
{

```

```

        if (id == null) return NotFound();

        var userId = _userManager.GetUserId(User);
        var favourite = await _context.DbFavourite
            .FirstOrDefaultAsync(f => f.Id == id && f.UserId
== userId);
        if (favourite == null) return NotFound();

        return View(favourite);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id, Favourite
editedFav)
    {
        var userId = _userManager.GetUserId(User);
        var favourite = await _context.DbFavourite
            .FirstOrDefaultAsync(f => f.Id == id && f.UserId
== userId);
        if (favourite == null)
            return NotFound();
        favourite.Comment = editedFav.Comment;
        _context.DbFavourite.Update(favourite);
        await _context.SaveChangesAsync();

        return RedirectToAction(nameof(Index));
    }

    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null) return NotFound();

        var userId = _userManager.GetUserId(User);
        var favourite = await _context.DbFavourite
            .Include(f => f.Offer)
            .FirstOrDefaultAsync(m => m.Id == id && m.UserId
== userId);

        if (favourite == null) return NotFound();

        return View(favourite);
    }

    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        var userId = _userManager.GetUserId(User);

        var favourite = await _context.DbFavourite
            .FirstOrDefaultAsync(f => f.Id == id && f.UserId
== userId);

```

```

        if (favourite != null)
        {
            _context.DbFavourite.Remove(favourite);
            await _context.SaveChangesAsync();
        }

        return RedirectToAction(nameof(Index));
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> AddAjax([FromBody]
FavouriteDto model)
    {
        var user = await _userManager.GetUserAsync(User);
        if (user == null)
        {
            return Json(new { success = false, message =
"Пользователь не найден." });
        }
        var alreadyExists = await _context.DbFavourite
            .AnyAsync(f => f.UserId == user.Id && f.OfferId ==
model.OfferId);
        if (!alreadyExists)
        {
            var favourite = new Favourite
            {
                OfferId = model.OfferId,
                UserId = user.Id
            };
            _context.DbFavourite.Add(favourite);
            await _context.SaveChangesAsync();
        }
        return Json(new { success = true });
    }

    public class FavouriteDto
    {
        public int OfferId { get; set; }
    }
}

```

Листинг Б.3 – Файл Index.cshtml

```

@model MortgageCalc.Models.MortgageCalculatorViewModel
@Html.AntiForgeryToken()
@{
    ViewData["Title"] = "Ипотечный калькулятор";
}

<link rel="stylesheet" href="~/css/mortgage.css" />

```

```

<link
href="https://cdn.jsdelivr.net/npm/sweetalert2@11/dist/sweetalert2
.min.css" rel="stylesheet">
<script
src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
<section class="calculator-header">
    <div class="row align-items-start">
        <div class="col-md-10">
            <p class="calc-subtext">
                Расчет и подбор оптимального варианта,
сравнение предложений разных банков. Доступен выбор 30 <br />
                предложений в 20 банках с первоначальным
взносом от 10 % и процентными ставками от 4,9% до <br />
                25,99%. Сделай первый шаг к своей мечте!
            </p>
        </div>
    </div>
</section>
<div class="calculator-container">
    <form asp-action="Index" method="post" class="calculator-
form">

        <div class="form-row">
            <label>Стоимость жилья:</label>
            <input id="propertyPrice"
                type="number" name="propertyPrice"
                min="0" step="0.01" required />
        </div>

        <div class="form-row">
            <label>
                Первоначальный взнос:
                <span id="dpPercent" class="text-muted">(0
%)</span>
            </label>
            <input id="initialPayment"
                type="number" name="initialPayment"
                min="0" step="0.01" required />
        </div>

        <div class="form-row">
            <label>Срок кредита (лет):</label>
            <div class="term-wrapper">
                <input type="number"
                    id="loanTermInput"
                    name="loanTerm"
                    min="1"
                    max="40"
                    value="20"
                    required />
                <input type="range"
                    id="loanTermSlider"
                    min="1"

```

```

        max="40"
        step="1"
        value="20" />
    </div>
</div>

<div class="form-row">
    <label>Процентная ставка (%):</label>
    <div class="interest-wrapper">
        <input type="number"
            id="interestInput"
            name="interestRate"
            step="0.1"
            min="4"
            max="25"
            value="16.5" />
        <input type="range"
            id="interestSlider"
            step="0.1"
            min="4"
            max="25"
            value="16.5" />
    </div>
</div>

<div class="form-row">
    <label>Дата начала выплат:</label>
    <input type="date" name="startDate" required />
</div>

<div class="form-row">
    <label>Тип платежа:
        <span class="info-tooltip" tabindex="0">
            <i class="fas fa-circle-question"></i>
            <span class="tooltip-text">
                <b>Аннуитетный платёж</b> – одинаковый
каждый месяц, сначала преимущественно проценты.<br><br>
                <b>Дифференцированный платёж</b> –
уменьшается со временем: основной долг делится равномерно,
проценты – по остаточной сумме.
            </span>
        </span>
    </label>
    <select name="paymentType">
        <option value="annuity">Аннуитетный</option>
        <option
value="differentiated">Дифференцированный</option>
    </select>
</div>

<button type="submit" class="calculate-
button">Рассчитать</button>

```

```

</form>

@if (Model != null && Model.PaymentSchedule != null &&
Model.PaymentSchedule.Any())
{
    <div class="result-section">
        <h2>Результаты расчёта</h2>
        <div class="results-container">
            <div class="result-grid">
                <div class="result-item"><span>Ежемесячный
платёж:</span><strong>@Model.MonthlyPayment.ToString("C")</strong></div>
                <div class="result-item"><span>Сумма
кредита:</span><strong>@Model.CreditAmount.ToString("C")</strong></div>
                <div class="result-
item"><span>Переплата:</span><strong>@Model.Overpayment.ToString("
C")</strong></div>
                <div class="result-item"><span>Общая
выплата:</span><strong>@Model.TotalPayment.ToString("C")</strong></div>
                <div class="result-item"><span>Необходимый
доход:</span><strong>@Model.RequiredIncome.ToString("C")</strong></div>
            </div>
            <div class="chart-wrapper">
                <div class="chart-legend">
                    <div>
                        <div class="legend-color loan-
color"></div>
                        <div>
                            <span class="legend-
label">Кредит</span><br />
<strong>@Model.CreditAmount</strong>
                        </div>
                    </div>
                    <div>
                        <div class="legend-color interest-
color"></div>
                        <div>
                            <span class="legend-
label">Проценты</span><br />
<strong>@ (Model.TotalPayment -
Model.CreditAmount)</strong>
                        </div>
                    </div>
                </div>
                <canvas id="mortgageChart"
                    data-loan="@Model?.CreditAmount"
                    data-total="@Model?.TotalPayment">
                </canvas>
            </div>
        </div>
    </div>
}

```

```

</div>

<h3>График платежей</h3>
<div class="d-flex gap-2 mt-3">
    <button id="toggleSchedule" class="btn btn-
outline-primary">Показать график платежей</button>

    <form asp-action="DownloadExcel"
method="post">

        <button type="submit" class="btn btn-outline-
success">📄 Скачать график</button>
    </form>

</div>
<div id="schedule-container" style="display: none;
max-height: 400px; overflow-y: auto; border: 1px solid #ccc;
border-radius: 5px; margin-top: 1rem;">
    <table class="payment-schedule table table-sm
table-bordered">
        <thead class="table-light">
            <tr>
                <th>№</th>
                <th>Дата платежа</th>
                <th>Сумма платежа</th>
                <th>Платёж по основному долгу</th>
                <th>Платёж по процентам</th>
                <th>Остаток долга</th>
            </tr>
        </thead>
        <tbody>
            @for (int i = 0; i <
Model.PaymentSchedule.Count; i++)
            {
                <tr>
                    <td>@(i + 1)</td>

                    <td>@Model.PaymentSchedule[i].PaymentDate.ToString("dd.MM.yyyy")</
td>

                    <td>@Model.PaymentSchedule[i].PaymentAmount.ToString("C")</td>

                    <td>@Model.PaymentSchedule[i].PrincipalPayment.ToString("C")</td>

                    <td>@Model.PaymentSchedule[i].InterestPayment.ToString("C")</td>

                    <td>@Model.PaymentSchedule[i].RemainingDebt.ToString("C")</td>
                </tr>
            }
        </tbody>
    </table>
</div>

```

```

</div>

<div class="offers-section">
    <h2>Ипотечные программы для вас</h2>
    @if (Model.Offers != null && Model.Offers.Any())
    {
        <div class="offers-grid">
            @foreach (var offer in Model.Offers)
            {
                <div class="offer-card">
                    <div class="offer-header">
                        <div>
                            <h5 class="bank-
name">@offer.BankTitle</h5>
                            <h4 class="offer-
title">@offer.Title</h4>
                        </div>
                    </div>
                    <div class="offer-info">
                        <p><strong>Процентная
ставка:</strong> @offer.InterestRate.ToString("0.##")%</p>
                        <p>
                            <strong>Полная стоимость
кредита:</strong>
                            <span class="info-tooltip"
tabindex="0">
                                <i class="fas fa-circle-
question"></i>
                                <span class="tooltip-
text">
                                    Полная стоимость
кредита — это фактическая процентная ставка, включающая все
обязательные платежи: проценты, комиссии, страхование и т.д.
                                </span>
                            </span>
                        <p>@offer.LoanCostMin.ToString()% -
@offer.LoanCostMax.ToString()%</p>
                    </div>
                    @if (User.Identity.IsAuthenticated)
                    {
                        bool isFavourite =
Model.FavouriteOfferIds.Contains(offer.Id);
                        <button class="btn btn-outline-danger
fav-btn"
                                data-offer-id="@offer.Id"
                                title="@ (isFavourite ?
"Добавлено в избранное" : "Добавить в избранное")"
                                @(isFavourite ? "disabled" : "")>
                            @(isFavourite ? "❤️" : "💔")
                        </button>
                    }
                }
            }
        </div>
    }
</div>

```

```

        {
            <button type="button" class="btn btn-
outline-danger unauth-fav-btn" title="Добавить в
избранное">♥</button>
        }
        <a href="@offer.Link" class="offer-
link" target="_blank">Перейти на сайт банка</a>
    </div>
    }
</div>
}
else if (ViewBag.NoOffersMessage != null)
{
    <p>@ViewBag.NoOffersMessage</p>
}
</div>
}
<section class="faq-section">
    <div class="faq-container">
        <h2>Часто задаваемые вопросы</h2>

        <div class="faq-item">
            <button class="faq-question">Как рассчитывается
ежемесячный платёж?</button>
            <div class="faq-answer">
                <p>Он рассчитывается с учётом введённой суммы
кредита, срока, процентной ставки и типа платежа (аннуитетный или
дифференцированный).</p>
            </div>
        </div>

        <div class="faq-item">
            <button class="faq-question">Что такое полная
стоимость кредита?</button>
            <div class="faq-answer">
                <p>Это эффективная ставка, включающая
проценты, обязательные комиссии, страхование и т.д. Она отражает
реальные расходы по кредиту.</p>
            </div>
        </div>

        <div class="faq-item">
            <button class="faq-question">Можно ли изменить
ставку вручную?</button>
            <div class="faq-answer">
                <p>Да. Используйте ползунок или введите точное
значение в поле «Процентная ставка».</p>
            </div>
        </div>

        <div class="faq-item">
            <button class="faq-question">Почему не
отображаются предложения от банков?</button>

```

```

        <div class="faq-answer">
            <p>Возможно, параметры кредита не
соответствуют условиям ни одного предложения. Попробуйте изменить
сумму, срок или взнос.</p>
        </div>
    </div>
</div>
</section>

</div>
@section Scripts {
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script>
        const chartCanvas =
document.getElementById('mortgageChart');
        const loanAmount = parseFloat(chartCanvas.dataset.loan);
        const totalPayment =
parseFloat(chartCanvas.dataset.total);

        if (!isNaN(loanAmount) && !isNaN(totalPayment)) {
            const interestPaid = totalPayment - loanAmount;

            new Chart(chartCanvas, {
                type: 'doughnut',
                data: {
                    labels: ['Кредит', 'Проценты'],
                    datasets: [{
                        data: [loanAmount, interestPaid],
                        backgroundColor: ['#355E3B', '#B9D9A5'],
                        borderWidth: 2
                    }]
                },
                options: {
                    responsive: false,
                    maintainAspectRatio: false,
                    cutout: '70%',
                    plugins: {
                        legend: { display: false },
                        tooltip: {
                            callbacks: {
                                label: function (context) {
                                    return context.label + ': ' +
context.formattedValue + ' ₴';
                                }
                            }
                        }
                    }
                }
            });
        }
    </script>
    <script>

```

```

document.getElementById("toggleSchedule").addEventListener("click"
, function () {
    const container = document.getElementById("schedule-
container");
    const isVisible = container.style.display === "block";
    container.style.display = isVisible ? "none" :
"block";
    this.textContent = isVisible ? "Показать график
платежей" : "Скрыть график платежей";
});

```

```

document.getElementById("downloadSchedule").addEventListener("clik
k", function () {
    const rows = [...document.querySelectorAll(".payment-
schedule tr")];
    let csvContent = "";

    rows.forEach(row => {
        const cols = [...row.querySelectorAll("th, td")];
        const line = cols.map(col =>
`"${col.innerText.replace(/\s+/g, ' ').trim()}"`).join(";");
        csvContent += line + "\n";
    });

```

```

    const blob = new Blob([csvContent], { type:
"text/csv;charset=utf-8;" });
    const url = URL.createObjectURL(blob);
    const link = document.createElement("a");
    link.setAttribute("href", url);
    link.setAttribute("download",
"mortgage_schedule.csv");
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
});

```

```

</script>

```

```

<script>

```

```

    document.querySelectorAll('.unauth-fav-
btn').forEach(button => {
        button.addEventListener('click', function () {
            Swal.fire({
                title: 'Войдите или зарегистрируйтесь',
                html: 'Чтобы добавить предложение в избранное,
необходимо <b>войти</b> или <b>создать аккаунт</b>.',
                icon: 'info',
                showCancelButton: true,
                showDenyButton: true,
                confirmButtonText: 'Войти',
                denyButtonText: 'Регистрация',
                cancelButtonText: 'Отмена',
                confirmButtonColor: '#3085d6',

```

```

        denyButtonColor: '#28a745',
        cancelButtonColor: '#d33'
    }).then((result) => {
        if (result.isConfirmed) {
            window.location.href =
'/Identity/Account/Login';
        } else if (result.isDenied) {
            window.location.href =
'/Identity/Account/Register';
        }
    });
});
});
document.querySelectorAll('.fav-btn').forEach(button => {
    button.addEventListener('click', function () {
        const offerId = this.dataset.offerId;
        const button = this;

        fetch('/Favourites/AddAjax', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'RequestVerificationToken':
document.querySelector('input[name="__RequestVerificationToken"]')
.value
            },
            body: JSON.stringify({ offerId: offerId })
        })
        .then(response => {
            if (!response.ok) throw new Error("Ошибка при
добавлении в избранное.");
            return response.json();
        })
        .then(data => {
            if (data.success) {
                button.innerText = '❤️';
                button.disabled = true;
                button.title = "Добавлено в избранное";
            } else {
                throw new Error(data.message || "Не
удалось добавить.");
            }
        })
        .catch(error => {
            console.error("Ошибка:", error);
            Swal.fire("Ошибка", error.message, "error");
        });
    });
});
});
</script>
<script>

```

```

    const interestInput =
document.getElementById('interestInput');
    const interestSlider =
document.getElementById('interestSlider');

    function syncInterestSlider(value) {
        interestSlider.value = value;
        const progress = (value - interestSlider.min) /
(interestSlider.max - interestSlider.min) * 100;
        interestSlider.style.setProperty('--range-progress',
`${progress}%`);
    }

    interestInput.addEventListener('input', () => {
        const val = parseFloat(interestInput.value);
        if (val >= 1 && val <= 100) syncInterestSlider(val);
    });

    interestSlider.addEventListener('input', () => {
        interestInput.value = interestSlider.value;
        syncInterestSlider(interestSlider.value);
    });

    syncInterestSlider(interestInput.value);

    const loanTermInput =
document.getElementById('loanTermInput');
    const loanTermSlider =
document.getElementById('loanTermSlider');

    function syncLoanSlider(value) {
        loanTermSlider.value = value;
        const progress = (value - loanTermSlider.min) /
(loanTermSlider.max - loanTermSlider.min) * 100;
        loanTermSlider.style.setProperty('--range-progress',
`${progress}%`);
    }

    loanTermInput.addEventListener('input', () => {
        const val = parseInt(loanTermInput.value);
        if (val >= 1 && val <= 75) syncLoanSlider(val);
    });

    loanTermSlider.addEventListener('input', () => {
        loanTermInput.value = loanTermSlider.value;
        syncLoanSlider(loanTermSlider.value);
    });

    syncLoanSlider(loanTermInput.value);

</script>

```

```

<script>
    const priceInput =
document.getElementById('propertyPrice');
    const downInput =
document.getElementById('initialPayment');
    const percentLabel = document.getElementById('dpPercent');
    const DEFAULT_RATE = 0.20;

    priceInput.addEventListener('input', () => {
        const price = parseFloat(priceInput.value);
        if (!isFinite(price) || price <= 0) {
            downInput.value = '';
            percentLabel.textContent = '(0 %)';
            return;
        }

        const dp = +(price * DEFAULT_RATE).toFixed(2);
        downInput.value = dp;

        percentLabel.textContent = '(20 %)';
    });

    downInput.addEventListener('input', () => {
        const price = parseFloat(priceInput.value);
        const down = parseFloat(downInput.value);

        if (!isFinite(price) || price <= 0 || !isFinite(down)
|| down < 0) {
            percentLabel.textContent = '(0 %)';
            return;
        }
        const pct = (down / price) * 100;
        percentLabel.textContent = `(${pct.toFixed(1)} %)`;
    });
</script>
<script>
    document.querySelectorAll('.faq-question').forEach(button
=> {
        button.addEventListener('click', () => {
            const item = button.parentElement;
            item.classList.toggle('active');
        });
    });
</script>
}

```

Листинг Б.4 – Файл Guide.cshtml

```

@{
    ViewData["Title"] = "Справка по калькулятору";
}

<link rel="stylesheet" href="~/css/mortgage.css" />

```

```

<section class="help-section">
  <div class="container">
    <div class="help-card">
      <h1 class="help-title">Как пользоваться ипотечным
калькулятором 📊</h1>

      <p>Этот калькулятор поможет вам рассчитать примерный
ежемесячный платёж, переплату, необходимый доход и выбрать
оптимальную ипотечную программу.</p>

      <h3>Пошаговая инструкция:</h3>
      <ol class="help-list">
        <li><strong>Стоимость жилья</strong> – введите
стоимость объекта недвижимости.</li>
        <li><strong>Первоначальный взнос</strong> –
укажите сумму или выберите процент от стоимости.</li>
        <li><strong>Срок кредита</strong> – укажите
желаемый срок погашения ипотеки в годах.</li>
        <li><strong>Процентная ставка</strong> –
используйте ползунок или ручной ввод для указания ставки.</li>
        <li><strong>Дата начала выплат</strong> – выберите
дату, с которой начнётся погашение.</li>
        <li><strong>Тип платежа</strong> – выберите
<em>аннуитетный</em> или <em>дифференцированный</em> платёж:</li>
      </ol>

      <ul class="highlight-box">
        <li>📌 <strong>Аннуитетный</strong> – ежемесячный
платёж остаётся постоянным.</li>
        <li>📉 <strong>Дифференцированный</strong> –
платёж уменьшается с каждым месяцем.</li>
      </ul>

      <p>После ввода данных нажмите
<strong>«Рассчитать»</strong>, чтобы увидеть результаты:</p>

      <ul class="results-box">
        <li>💰 Ежемесячный платёж</li>
        <li>📊 Переплата</li>
        <li>📈 Необходимый доход</li>
        <li>📅 График платежей</li>
      </ul>

      <p>Ниже вы также увидите список актуальных ипотечных
программ от банков, соответствующих введённым параметрам.</p>

      <p class="mt-4">Если у вас есть вопросы, напишите нам
по электронному адресу: sofiamixailovna00@mail.ru</p>
    </div>
  </div>
</section>

```

Листинг Б.5 – Файл Details.cshtml

```
@model MortgageCalc.Models.Favourite

@{
    ViewData["Title"] = "Детали предложения";
}

<link rel="stylesheet" href="~/css/mortgage.css" />

<section class="details-section">
    <div class="details-card">
        <h1 class="details-title"> Детали предложения:
@Model.Offer.Title</h1>

        <div class="details-block">
            <h4><i class="fas fa-university me-2"></i>Информация о
банке</h4>
            <p><strong>Банк:</strong> @Model.Offer.Bank?.Title</p>
            <p><strong>Телефон:</strong>
@Model.Offer.Bank?.Phone</p>
            <p><strong>Адрес:</strong>
@Model.Offer.Bank?.Address</p>
            <p><strong>Лицензия:</strong>
@Model.Offer.Bank?.License</p>
        </div>

        <div class="details-block">
            <h4><i class="fas fa-chart-line me-2"></i>Основные
параметры</h4>
            <p><strong>Ставка:</strong> @Model.Offer.InterestRate
%</p>
            <p><strong>Стоимость жилья:</strong> от
@Model.Offer.LoanCostMin до @Model.Offer.LoanCostMax</p>
            <p><strong>Сумма кредита:</strong> от
@Model.Offer.LoanAmountMin до @Model.Offer.LoanAmountMax</p>
            <p><strong>Первоначальный взнос:</strong>
@Model.Offer.DownPayment</p>
            <p><strong>Срок кредита:</strong> @Model.Offer.TermMin
- @Model.Offer.TermMax лет</p>
            <p><strong>Стаж заемщика:</strong>
@Model.Offer.Experience лет</p>
            <p><strong>Ссылка:</strong> <a
href="@Model.Offer.Link" target="_blank">@Model.Offer.Link</a></p>
        </div>

        <div class="details-block">
            <h4><i class="fas fa-credit-card me-2"></i>Тип
платежа</h4>
            <p><strong>@Model.Offer.PaymentType?.Title</strong></p>
            <p>@Model.Offer.PaymentType?.Description</p>
        </div>
    </div>
</section>
```

```

</div>

<div class="details-block">
    <h4><i class="fas fa-file-alt me-2"></i>Дополнительные
условия</h4>

<p><strong>@Model.Offer.DataCondition?.Condition?.Title</strong></
p>

<p>@Model.Offer.DataCondition?.Condition?.Description</p>
    <p><strong>Начало действия:</strong>
@Model.Offer.DataCondition?.StartDate.ToString()</p>
    @if (Model.Offer.DataCondition?.EndDate != null)
    {
        <p><strong>Окончание действия:</strong>
@Model.Offer.DataCondition?.EndDate?.ToShortDateString()</p>
    }
</div>

<div class="details-block">
    <h4><i class="fas fa-comment-dots me-2"></i>Ваш
комментарий</h4>
    <p>@Model.Comment</p>
</div>

<div class="mt-4">
    <a href="javascript:history.back()" class="btn btn-
outline-secondary back-button">
        <i class="fas fa-arrow-left"></i> Назад
    </a>
</div>
</div>
</section>

```

Листинг Б.6 – Файл Login.cshtml

```

@page
@model LoginModel
@{
    ViewData["Title"] = "Вход";
}

<link rel="stylesheet" href="~/css/mortgage.css" />

<section class="auth-section">
    <div class="auth-card">
        <h1 class="auth-title">Вход в личный кабинет 🔒</h1>
        <p class="auth-subtext">Введите e-mail и пароль, указанные
при регистрации. Или войдите через соцсеть.</p>

        <form id="account" method="post">

```

```

        <div asp-validation-summary="ModelOnly" class="text-
danger mb-3" role="alert"></div>

        <div class="form-group">
            <label asp-for="Input.Email">Email</label>
            <input asp-for="Input.Email" class="form-control
custom-input" autocomplete="username" />
            <span asp-validation-for="Input.Email"
class="text-danger small"></span>
        </div>

        <div class="form-group">
            <label asp-for="Input.Password">Пароль</label>
            <input asp-for="Input.Password" class="form-
control custom-input" autocomplete="current-password" />
            <span asp-validation-for="Input.Password"
class="text-danger small"></span>
        </div>

        <div class="form-check mb-3">
            <input class="form-check-input" asp-
for="Input.RememberMe" />
            <label class="form-check-label" asp-
for="Input.RememberMe">Запомнить меня</label>
        </div>

        <button id="login-submit" type="submit" class="btn
btn-primary w-100 auth-submit-btn">Войти</button>

        <div class="mt-3 d-flex flex-column gap-2">
            <a asp-page="./Register" asp-route-
returnUrl="@Model.ReturnUrl" class="text-muted small">Нет
аккаунта? Зарегистрируйтесь</a>
        </div>
    </form>
</div>
</section>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

```

Листинг Б.7 – Файл PersonalData.cshtml

```

@page
@model PersonalDataModel
@{
    ViewData["Title"] = "Личные данные";
    ViewData["ActivePage"] = ManageNavPages.PersonalData;
}

<link rel="stylesheet" href="~/css/mortgage.css" />
<div class="personal-data-card">
    <p>

```

Ваша учетная запись содержит персональные данные, которые вы нам предоставили.

Вы можете **скачать** или **удалить** их.

`<p class="text-danger">`

⚠ Удаление этих данных приведет к безвозвратному удалению вашей учетной записи.

```

<div class="d-flex flex-column flex-sm-row align-items-
stretch gap-3 mt-4">
  <div class="personaldata-bat">
    <form id="download-data" asp-
page="DownloadPersonalData" method="post" class="w-100">
      <button class="btn btn-outline-success w-100 py-2
d-flex align-items-center justify-content-center" type="submit">
        <i class="fas fa-download me-2"></i> Скачать
данные
      </button>
    </form>
  </div>
  <div class="personaldata-batdel">
    <a id="delete" asp-page="DeletePersonalData"
      class="btn btn-outline-danger w-100 py-2 d-flex
align-items-center justify-content-center">
      <i class="fas fa-trash-alt me-2"></i> Удалить
аккаунт
    </a>
  </div>
</div>
</div>
@section Scripts {
  <partial name="_ValidationScriptsPartial" />
}

```

Приложение В
(рекомендуемое)
Графическая часть

Использование ресурсов от Вт 13.05.25
Проект1.mpp

	Итого
Аналитик	104 ч
Анализ назначения и условий применения объектов предметной области	8 ч
Анализ приложений для расчета ипотечных кредитов	8 ч
Анализ функциональных требований	8 ч
Анализ нефункциональных требований	8 ч
Планирование процесса разработки приложения и оценка затрат на разработку	8 ч
Оформление пояснительной записки	64 ч
Программист	172 ч
Проектирование архитектура приложения	24 ч
Проектирование классов программных средств	16 ч
Реализация основного приложения	48 ч
Реализация интерфейса приложения	20 ч
Оформление пояснительной записки	64 ч
Дизайнер	36 ч
Проектирование пользовательского интерфейса	16 ч
Реализация интерфейса приложения	20 ч
Разработчик базы данных	32 ч
Проектирование структур данных	16 ч
Реализация базы данных	16 ч
Тестировщик	48 ч
Выбор режима тестирования и метрик кода	8 ч
Тестирование пользовательской части приложения для анализа условий и подбора ипотечных предложений	16 ч
Тестирование серверной части приложения для анализа условий и подбора ипотечных предложений	16 ч
Оценка качества программного кода	8 ч
Компьютер	
Электричество	
Анализ назначения и условий применения объектов предметной области	
Анализ приложений для расчета ипотечных кредитов	
Анализ функциональных требований	
Анализ нефункциональных требований	
Планирование процесса разработки приложения и оценка затрат на разработку	
Проектирование архитектура приложения	
Проектирование структур данных	
Проектирование пользовательского интерфейса	
Проектирование классов программных средств	
Реализация базы данных	
Реализация основного приложения	
Реализация интерфейса приложения	
Выбор режима тестирования и метрик кода	
Тестирование пользовательской части приложения для анализа условий и подбора ипотечных предложений	
Тестирование серверной части приложения для анализа условий и подбора ипотечных предложений	
Оценка качества программного кода	
Оформление пояснительной записки	
Интернет	
Анализ назначения и условий применения объектов предметной области	
Анализ приложений для расчета ипотечных кредитов	
Анализ функциональных требований	
Анализ нефункциональных требований	
Проектирование архитектура приложения	
Проектирование пользовательского интерфейса	
Реализация базы данных	
Реализация основного приложения	
Реализация интерфейса приложения	
Выбор режима тестирования и метрик кода	
Тестирование пользовательской части приложения для анализа условий и подбора ипотечных предложений	
Тестирование серверной части приложения для анализа условий и подбора ипотечных предложений	
Итого	392 ч

Рисунок В.1 – Отчет об использовании ресурсов

Отчет о бюджете от Вт 13.05.25
Проект1.mpp

Ид.	Название задачи	Фиксированные затраты	Начисление фикс. затрат	Общие затраты	Базовые	Отклонение
27	Оформление пояснительной записки	0,00 Р	Пропорциональное	32 200,00 Р	0,00 Р	32 200,00 Р
17	Реализация основного приложения	0,00 Р	Пропорциональное	14 640,00 Р	0,00 Р	14 640,00 Р
18	Реализация интерфейса приложения	0,00 Р	Пропорциональное	10 200,00 Р	0,00 Р	10 200,00 Р
10	Проектирование архитектуры приложения	0,00 Р	Пропорциональное	7 320,00 Р	0,00 Р	7 320,00 Р
13	Проектирование классов программ	0,00 Р	Пропорциональное	4 850,00 Р	0,00 Р	4 850,00 Р
16	Реализация базы данных	0,00 Р	Пропорциональное	4 080,00 Р	0,00 Р	4 080,00 Р
11	Проектирование структур данных	0,00 Р	Пропорциональное	4 050,00 Р	0,00 Р	4 050,00 Р
12	Проектирование пользовательского интерфейса	0,00 Р	Пропорциональное	3 280,00 Р	0,00 Р	3 280,00 Р
22	Тестирование пользовательской части	0,00 Р	Пропорциональное	3 280,00 Р	0,00 Р	3 280,00 Р
23	Тестирование серверной части приложения	0,00 Р	Пропорциональное	3 280,00 Р	0,00 Р	3 280,00 Р
3	Анализ назначения и условий применения	0,00 Р	Пропорциональное	1 640,00 Р	0,00 Р	1 640,00 Р
4	Анализ приложений для расчета ипс	0,00 Р	Пропорциональное	1 640,00 Р	0,00 Р	1 640,00 Р
5	Анализ функциональных требований	0,00 Р	Пропорциональное	1 640,00 Р	0,00 Р	1 640,00 Р
6	Анализ нефункциональных требований	0,00 Р	Пропорциональное	1 640,00 Р	0,00 Р	1 640,00 Р
21	Выбор режима тестирования и метр	0,00 Р	Пропорциональное	1 640,00 Р	0,00 Р	1 640,00 Р
7	Планирование процесса разработки	0,00 Р	Пропорциональное	1 625,00 Р	0,00 Р	1 625,00 Р
24	Оценка качества программного кода	0,00 Р	Пропорциональное	1 625,00 Р	0,00 Р	1 625,00 Р
1	Начало проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
8	Анализ предметной области и выяв.	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
14	Проектирование приложения-агрегатора	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
19	Реализация приложения-агрегатора	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
25	Контроль качества программного обеспечения	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
28	Оформление отчета завершено	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
29	Конец проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
		0,00 Р		98 630,00 Р	0,00 Р	98 630,00 Р

Рисунок В.2 – Отчет о бюджете проекта

Приложение Г
(рекомендуемое)
Результаты функционального тестирования

Стоимость жилья:

1000

Срок кредита ! Значение должно быть больше 1000000

Рисунок Г.1 – Результаты тестирования класса эквивалентности «Длина меньше семи символов»

Стоимость жилья:

1|

Срок кредита (лет):

Рисунок Г.2 – Результаты тестирования класса эквивалентности «Недопустимость ввода букв»

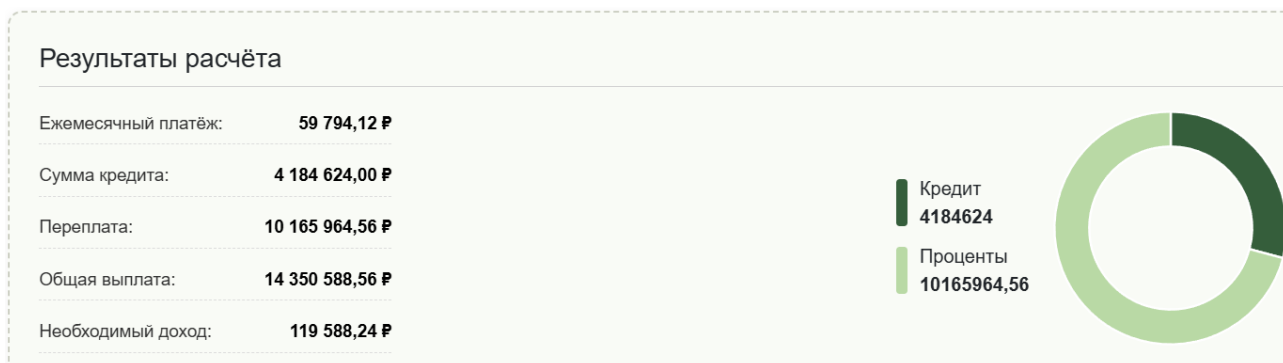


Рисунок Г.3 – Результаты тестирования класса эквивалентности «Первая цифра ноль»

Стоимость жилья:

01-

Срок кредита (лет):

Введите число.

Рисунок Г.4 – Результаты тестирования класса эквивалентности «Длина меньше семи, содержит не только цифры, начинается с нуля»

Стоимость жилья:

0-09876|

Срок кредита (лет):

Введите число.

Рисунок Г.5 – Результаты тестирования класса эквивалентности «Длина больше девяти содержит не только цифры, начинается с нуля»

Стоимость жилья:

Срок кредита (лет):

Вы пропустили это поле.

Рисунок Г.6 – Результаты тестирования логической границы «Отсутствие символов»

Стоимость жилья:

-1000000

Срок кредита:

Значение должно быть больше или равно 1000

Рисунок Г.7 – Результаты тестирования логической границы «Отрицательное число»

Скриншот веб-формы с двумя полями: «Стоимость жилья» и «Срок кредита». Поле «Стоимость жилья» содержит значение 0. Поле «Срок кредита» содержит значение 1000. Над полем «Срок кредита» отображается сообщение об ошибке: «Значение должно быть больше или равно 1000».

Стоимость жилья:

0

Срок кредита: ! Значение должно быть больше или равно 1000

Рисунок Г.8 – Результаты тестирования логической границы «Число равно нулю и начинается с нуля»

Скриншот веб-формы с двумя полями: «Стоимость жилья» и «Срок кредита». Поле «Стоимость жилья» содержит значение 4312000,58. Поле «Срок кредита» содержит значение 1000. Над полем «Срок кредита» отображается сообщение об ошибке: «Значение должно быть целым».

Стоимость жилья:

4312000,58

Срок кредита: ! Значение должно быть целым

Рисунок Г.9 – Результаты тестирования логической границы «Число не является целым»

Скриншот веб-формы с двумя полями: «Стоимость жилья» и «Срок кредита». Поле «Стоимость жилья» содержит значение 893248796201599785200598. Поле «Срок кредита» содержит значение 1000. Над полем «Срок кредита» отображается сообщение об ошибке: «Значение должно быть меньше».

Стоимость жилья:

893248796201599785200598

Срок кредита: ! Значение должно быть меньше

Рисунок Г.10 – Результаты тестирования логической границы «Размер числа больше допустимого по типу»

Приложение Д
(рекомендуемое)
Листинг модульных тестов

```

using System;
using System.Collections.Generic;
using Xunit;
using MortgageCalculatorApp;

public class MortgageCalculatorTests
{
    [Fact]
    public void CalculateLoanAmount_CorrectValue()
    {
        Assert.Equal(400000,
MortgageCalculator.CalculateLoanAmount(500000, 100000));
        Assert.Equal(500000,
MortgageCalculator.CalculateLoanAmount(700000, 200000));
    }

    [Fact]
    public void CalculateMonthlyPayment_CorrectValue()
    {
        Assert.Equal(4824.32,
MortgageCalculator.CalculateMonthlyPayment(500000, 10, 20), 2);
        Assert.Equal(2866.12,
MortgageCalculator.CalculateMonthlyPayment(300000, 8, 15), 2);
    }

    [Fact]
    public void CalculateTotalPayment_CorrectValue()
    {
        Assert.Equal(1157836.80,
MortgageCalculator.CalculateTotalPayment(4824.32, 20), 2);
        Assert.Equal(515901.60,
MortgageCalculator.CalculateTotalPayment(2866.12, 15), 2);
    }

    [Fact]
    public void CalculateRequired_CorrectValue()
    {
        Assert.Equal(16081.07,
MortgageCalculator.CalculateRequiredIncome(4824.32, 30), 2);
        Assert.Equal(11464.48,
MortgageCalculator.CalculateRequiredIncome(2866.12, 25), 2);
    }

    [Fact]
    public void GetBankOffers_ValidOffers()
    {
        var offers = MortgageCalculator.GetBankOffers(500000, 10,
20);
        Assert.NotNull(offers);
        Assert.NotEmpty(offers);
        Assert.All(offers, offer =>
Assert.NotNull(offer.BankName));
    }
}

```

```
        Assert.All(offers, offer => Assert.True(offer.InterestRate
> 0));
        Assert.All(offers, offer =>
Assert.True(offer.MonthlyPayment > 0));
    }
}
```