МО Аларский район

МБОУ Кутуликская СОШ

**Творческий проект**

«ИНФОГРАФИКА. ИПОТЕЧНЫЙ КАЛЬКУЛЯТОР»

**Направление проекта:** Информационный

**Вид проекта:** Практико-ориентированный

Выполнила: Ученицы 10класса

Иванова Яна и Немчинова Кристина

Руководитель: Учитель математики и информатики

Прокопьев Алексей Анатольевич

2025 год

Оглавление:

[ВВЕДЕНИЕ 3](#_Toc193906457)

[КОНФИГУРАЦИЯ И СРЕДА РАЗРАБОТКИ 4](#_Toc193906458)

[РАЗРАБОТКА ПРИЛОЖЕНИЯ 5](#_Toc193906459)

[СЕРВЕРНАЯ ЛОГИКА 7](#_Toc193906460)

[ДЕПЛОЙ И СБОРКА 7](#_Toc193906461)

[ВИЗУАЛЬНАЯ СЛОЙ И ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС 8](#_Toc193906462)

[ИНТЕГРАЦИЯ И ВЗАИМОДЕЙСТВИЕ ЧАСТЕЙ ПРОЕКТ 9](#_Toc193906463)

[ДОПОЛНИТЕЛЬНЫЕ АСПЕКТЫ И РАЗНООБРАЗИЕ ФУНКЦИЙ 9](#_Toc193906464)

[СЦЕНАРИИ РАБОТЫ И ПОЛЬЗОВАТЕЛЬСКИЙ ОПЫТ 10](#_Toc193906465)

[АЛГОРИТМЫ ВЫЧИСЛЕНИЯ И PAYLOAD ОБРАБОТКИ 11](#_Toc193906466)

[РОЛЬ ТЕСТИРОВАНИЯ И ВЕРИФИКАЦИИ 11](#_Toc193906467)

[ЭКОЛОГИЯ ПРОЕКТА И БУДУЩИЕ РАЗВИТИЯ 12](#_Toc193906468)

[ЗАКЛЮЧЕНИЕ 13](#_Toc193906469)

[СПИСОК ЛИТЕРАТУРЫ 14](#_Toc193906470)

# 

# ВВЕДЕНИЕ

Наш проект – это комплексное веб-приложение для расчёта ипотечных платежей, объединяющее современные технологии фронтенда, серверные вычисления, а также настольный запуск через Electron. Архитектура проекта разбита на отдельные подсистемы, каждая из которых отвечает за свою задачу: от настройки среды разработки до визуализации графиков платежей. Можно сказать, что проект построен по принципу «модульного офиса», где каждая комната (папка) имеет своё назначение, а все комнаты связаны между собой через общие сервисы и компоненты.

**ПРОБЛЕМА**: Существующие ипотечные калькуляторы часто перегружены сложной терминологией и требуют ввода избыточных данных, что затрудняет быстрое и точное планирование ипотечных выплат для обычного пользователя.

**ЦЕЛЬ ПРОЕКТА**: цель работы заключается в создании программы "Инфографика Ипотечный калькулятор", которая позволяет рассчитывать основные параметры ипотечного кредита, включая сумму переплаты, и представлять результаты в удобной визуальной форме.

**ЗАДАЧИ**:

1. Изучить основные принципы расчёта ипотечных платежей и переплаты по кредиту.

2. Разработать алгоритм программы, выполняющей необходимые вычисления.

3. Реализовать программное обеспечение с функцией расчёта параметров ипотеки.

4. Интегрировать элементы инфографики для наглядного представления результатов.

5. Провести тестирование программы и оценить её практическую применимость.

**СТРУКТУРА РАБОТЫ**: структура работы включает несколько разделов. Первый раздел рассматривает основы ипотечного кредитования и принципы инфографики. Второй раздел описывает разработку программы, включая алгоритмы и технические решения. Третий раздел содержит результаты, анализ функциональности и примеры визуализации. В заключении подводятся итоги, оценивается достижение цели и перспективы развития.

**АКТУАЛЬНОСТЬ ПРОЕКТА**: Программа "Инфографика Ипотечный калькулятор" упрощает расчёт и визуализацию параметров ипотеки, делая информацию доступной и повышая финансовую грамотность, что подчёркивает её актуальность и социальную значимость.

**СРОКИ ВЫПОЛНЕНИЯ РАБОТЫ:**

1. Подготовительный этап: 05.02.25 - 09.02.25
2. Основной этап: 10.02.25 – 03.03.25
   1. Разработка кода
   2. Создание теоретической части
3. Заключительный этап: 04.03.25 – 20.03.25
   1. Тестирование программы
   2. Реализация проекта

**ОЖИДАЕМЫЙ РЕЗУЛЬТАТ:**

Для пользователя:

• Предоставляет возможность быстро и точно рассчитать ежемесячные ипотечные платежи, используя минимальный набор необходимых и понятных параметров (сумма кредита, процентная ставка, срок, первоначальный взнос).

• Обеспечивает прозрачное представление информации о структуре платежей, общей стоимости и переплате по кредиту, избегая сложной терминологии и финансовых жаргонизмов.

• Позволяет обычному пользователю, не имеющему специальных финансовых знаний, самостоятельно и без затруднений планировать ипотечные выплаты и оценивать свои финансовые возможности.

• Сокращает время и усилия, необходимые для получения точной информации об ипотечных платежах.

• В целом:

• Повышает финансовую грамотность пользователей в вопросах ипотечного кредитования.

• Предоставляет пользователям инструмент для принятия более осознанных и обоснованных решений при выборе ипотечной программы.

• Улучшает пользовательский опыт за счет простоты и удобства использования.

КОНФИГУРАЦИЯ И СРЕДА РАЗРАБОТКИ

* tsconfig.json

Этот файл задаёт параметры для компиляции TypeScript. Он определяет библиотеки (DOM, esnext и т.д.), строгую проверку типов и перечень плагинов, включая плагин «next», который помогает интегрировать Next.js в среду разработки. Также здесь определены пути (алиасы) для упрощённого импорта файлов в проекте – например, alias "@/\*" ведёт к корневой папке проекта. Можно сравнить tsconfig.json с подробной инструкцией для сборки конструктора, где все детали (типы, модули, пути) заданы заранее.

* next.config.mjs

Конфигурационный файл Next.js задаёт общие параметры сборки и запуска. Здесь задаются:

– Игнорирование ошибок сборки при проверке типов или ESLint, чтобы ускорить процесс разработки.

– Параметры оптимизации изображений, а также экспериментальные настройки, такие как параллельные компиляции и сборка в воркерах.

– Функция mergeConfig объединяет стандартные конфигурационные параметры с опциональными пользовательскими настройками (файл v0-user-next.config), что позволяет гибко менять поведение приложения.

Этот файл можно уподобить бригаде инженеров, которая настраивает всю инфраструктуру перед запуском здания.

* package.json и package-lock.json

В package.json описаны зависимости проекта, набор скриптов для разработки, сборки и запуска Electron-приложения. Он задаёт «рецепт» для установки нужных модулей – будь то библиотеки для работы с React, Next.js, почтовыми уведомлениями, или Electron для создания нативного оконного интерфейса.

Package-lock.json фиксирует версии зависимостей, чтобы сборка всегда проводилась с одними и теми же библиотеками. Это гарантирует воспроизводимость сборки – как строгая спецификация материалов в строительной инструкции.

* Tailwind, PostCSS и компоненты стилей

Файлы tailwind.config.js и postcss.config.mjs отвечают за стилизацию интерфейса. Tailwind задаёт цвета, размеры, отступы и var–значения CSS-переменных. В конфигурации указан базовый цвет, расширенные настройки для контейнеров и кастомные значения для радиусов. PostCSS оборачивает процесс обработки CSS, чтобы Tailwind мог корректно работать с новым синтаксисом.

Файл components.json содержит сведения о том, как должны быть настроены UI-компоненты, включая алиасы для папок, базовые стили из Tailwind и выбор иконок из библиотеки lucide. Это как палитра и набор инструментов для художника, где каждый элемент интерфейса синхронизирован со стандартами всего проекта

# РАЗРАБОТКА ПРИЛОЖЕНИЯ

* Папка app: Главные страницы и глобальные стили

Папка app является «лицом» нашего веб-приложения. Здесь расположены:

– globals.css – файл, где задаются глобальные стили для всего приложения. Он определяет базовые правила для шрифтов, отступов и базовой цветовой палитры, позволяя интерфейсу выглядеть единообразно на всех страницах.

– layout.tsx – компонент, отвечающий за общее расположение страниц. Он задаёт шаблон для каждой страницы – шапка, основное содержимое, футер. Если сравнить с офисным зданием, это общий план этажей, по которому все комнаты и офисы спроектированы.

– page.tsx – конкретная страница приложения, которая может быть встроена в общий макет. Здесь происходит импорт основных компонентов, таких как калькулятор ипотеки, а также выполнение логики, отображающей результаты расчётов. В этой части пользователь взаимодействует с интерфейсом, как с главным залом консультациго центра.

Также в папке app может находиться отдельная подпапка api, реализующая API маршруты для клиентов, если подобная логика реализуется на стороне Next.js.

* Папка components: Визуальные и функциональные блоки

Эта папка содержит все React-компоненты, которые составляют пользовательский интерфейс приложения. Каждый компонент отвечает за свою задачу и взаимодействует с другими элементами через пропсы (набор входных параметров).

* bank-card.tsx

Компонент карточки банка. Он отображает краткую информацию об ипотечной программе банка – процентные ставки, сумму кредита, минимальный первоначальный взнос и дополнительные условия. Аналогией здесь можно назвать визитную карточку, которая быстро дает нужную информацию пользователю. При клике по карточке может происходить выбор программы и передача данных в форму расчёта.

* calculator-form.tsx

Компонент формы для ввода параметров ипотеки. Здесь пользователь заполняет поля для суммы кредита, процентной ставки, срока погашения и других деталей. В форме используется валидация данных с помощью React Hook Form. Например, для поля «сумма кредита» задаётся атрибут max, зависящий от ограничений выбранной программы. Это можно уподобить панели управления, где на дисплее видны ограничения и рекомендации для ввода данных.

* mortgage-charts.tsx

Компонент для визуализации результатов расчёта ипотеки. Используя библиотеки Recharts и framer-motion, он строит графики, диаграммы и анимированные блоки, которые показывают структуру платежей, остаток задолженности по месяцам и прогресс погашения кредита. Каждая часть графика снабжена подсказками (tooltips) и анимационными эффектами – как современное табло, информирующее пользователя о ходе расчётов.

Другие компоненты (например, custom-calculator.tsx, mortgage-results.tsx) отвечают за альтернативные варианты расчёта ипотеки и отображения подробных результатов. Их можно сравнить с разными рабочими станциями в офисе, где каждая отвечает за специализированную задачу.

* Папка data: Структурированные данные

Файл banks.ts из папки data содержит подробную информацию о банках и условиях ипотеки. Здесь для каждого банка описаны:

– Идентификатор и название банка.

– Логотип (URL или путь к изображению).

– Описание ипотечной программы: процентная ставка (minRate и maxRate), максимальная сумма кредита, минимальный первоначальный взнос, сроки кредита и список требований к заёмщику.

– Дополнительные сведения, такие как преимущества и ограничения программ (например, повышенная ставка при отсутствии страховки или условия для IT-специалистов).

Этот файл можно сравнить с каталогом банков, где каждая программа представлена как отдельный продукт с набором характеристик.

* Папка types: Типизация и контракты данных

Файл mortgage.ts содержит определения TypeScript-интерфейсов для данных, с которыми работает наш калькулятор. Интерфейсы описывают структуры объектов, такие как информации о банке, ипотечной программе и результатах расчёта. Это как подробный чертёж, по которому строятся детали – все компоненты ожидают данные именно в том формате, который описан здесь.

* Папка hooks: Пользовательские хуки

В этой папке находятся хуки, которые инкапсулируют повторяющуюся логику:

* use-mobile.tsx

Хук, определяющий особенности отображения и поведения на мобильных устройствах. Он может, например, переключать стили или изменять расположение элементов в зависимости от размера экрана.

* use-toast.ts

Отвечает за отображение уведомлений (тосты). Он позволяет централизованно оповещать пользователя о событиях (успешное выполнение расчёта, ошибки валидации и т.д.), как мини-сигналы внутри рабочего процесса.

Эти хуки позволяют избежать дублирования кода и сделать логику более декларативной.

* Папка lib: Вспомогательные функции

Файл utils.ts из папки lib содержит общие функции, используемые в разных частях приложения. Например, функция formatCurrency форматирует числа в денежный формат с учетом локальных настроек (например, добавляет разделители разрядов и валютный символ). Это можно сравнить с универсальным инструментом, который используется многими отделами для получения стандартного результата.

# СЕРВЕРНАЯ ЛОГИКА

* Папка api

Эта папка содержит серверный код для расчёта ипотеки. В частности, в папке api находится подпапка calculate, где файл route.py реализует маршрут API для вычислений. Хотя основное приложение написано на JavaScript/TypeScript (Next.js), для расчётов используется Python, что позволяет использовать дополнительные библиотеки для математических вычислений или работы с данными. Такой подход можно сравнить с тем, что основная витрина (фронтенд) работает на одном языке, а «кухня» (серверная логика) использует другой, специализированный язык для сложных вычислений.

* Виртуальное окружение (.venv)

Папка .venv содержит виртуальное окружение Python, в котором установлены необходимые пакеты для работы серверного кода. Это гарантирует, что серверные зависимости изолированы от системы и всегда выглядят одинаково вне зависимости от версии Python, установленной на машине разработчика. Такое окружение можно уподобить отдельному цеху, где все инструменты и материалы точно соответствуют заявленным стандартам.

# ДЕПЛОЙ И СБОРКА

* main.js и Electron

Файл main.js в корневой папке проекта отвечает за запуск Electron-приложения. Он создаёт окно с заданными размерами (800×600) и загружает URL, на котором работает веб-приложение (например, http://localhost:3000). Логика обработки событий (закрытие окна, повторное создание окна при активации) позволяет сделать приложение похожим на нативное – как если бы офисное окно запускалось операционной системой, а не браузером.

* Папка dist

Папка dist предназначена для выходных файлов, которые генерируются после сборки проекта. Здесь находятся собранные артефакты, оптимизированные для распространения на разных платформах. В данной папке могут быть итоговые сборки для Electron-версии приложения – это как финальные макеты, готовые к печати или распространению.

# ВИЗУАЛЬНАЯ СЛОЙ И ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

* Глобальные стили и компоненты UI

Глобальные стили, задаваемые в файлах globals.css (из папок app и styles), определяют внешний вид и ощущения от приложения. Они задают основы для всех компонентов, как если бы весь офис был оформлен в едином стиле, с едиными шрифтами, цветами и отступами. При этом использованы современные CSS-подходы с использованием переменных, что позволяет легко менять тему (например, переключать на тёмный режим).

Компоненты пользовательского интерфейса (например, Card, Badge) реализованы с использованием библиотеки radix-ui или собственных решений. Это гарантирует единообразие и гибкость при построении различных страниц, от карточек банков до детализированных форм расчёта. Каждый компонент обладает своей логикой и визуальными эффектами – от анимации до взаимодействия с пользователем (нажатия, hover-эффекты).

* Визуализация данных через charts

Компонент mortgage-charts.tsx отвечает за построение диаграмм и графиков. С помощью Recharts и дополнительных библиотек (framer-motion) создаются динамичные и отзывчивые визуализации. Здесь можно увидеть следующие элементы:

– Круговые диаграммы, иллюстрирующие распределение суммы кредита между процентной ставкой, основным долгом и страховкой.

– Линейные графики, отображающие остаток задолженности по месяцам, с подробными аннотациями (например, месяц расчёта и процент выполнения плана).

– Анимированные элементы, такие как прогресс-бары, показывающие процент выполнения кредита.

Эти визуальные элементы позволяют пользователю получить наглядную картину финансовых потоков – как если бы на офисном табло отображались все ключевые показатели.

# ИНТЕГРАЦИЯ И ВЗАИМОДЕЙСТВИЕ ЧАСТЕЙ ПРОЕКТ

Проект построен на принципах модульности и взаимосвязи компонентов. Рассмотрим, как отдельные элементы взаимодействуют между собой.

* Отправка данных пользователя и расчёт результатов

При вводе данных в форму калькулятора (calculator-form.tsx) данные сразу проходят валидацию с помощью React Hook Form и, в зависимости от выбранной банковской программы (из banks.ts), применяются ограничения (минимальное/максимальное значение суммы, процентной ставки и т.д.). Затем, при отправке формы, происходит вызов функции onCalculate, которая может отправить данные на API-роут (Python-скрипту) для расчёта ежемесячного платежа и общей суммы выплат.

Полученные сервером результаты возвращаются обратно в компонент mortgage-results.tsx, где происходит их форматирование (с помощью util-функций) и вывод на экран. Таким образом, весь процесс напоминает производственную линию: пользователь задаёт параметры, данные проходят через контроль качества (валидация), затем отправляются в цех вычислений, и на выходе получаем готовый продукт.

* Обновление визуальных элементов

После получения расчётов, данные переносятся в компонент mortgage-charts.tsx, который строит графики платежей. Используя состояние (useState) и анимационные библиотеки, диаграммы плавно отображаются на экране. Обновление UI происходит автоматически, как если бы дисплей в офисном центре обновлялся с новыми данными в режиме реального времени.

* Выбор программ и взаимодействие с карточками банков

В компоненте BankCard (bank-card.tsx) присутствует логика выбора ипотечной программы. При клике по карточке происходит вызов функции onSelect, которая передаёт выбранные параметры в родительский компонент mortgage-calculator.tsx. Таким образом, выбор карточки с конкретной программой банка сразу влияет на значения, показываемые в калькуляторе – аналогично тому, как консультант предлагает клиенту конкретный тариф после короткой беседы.

# ДОПОЛНИТЕЛЬНЫЕ АСПЕКТЫ И РАЗНООБРАЗИЕ ФУНКЦИЙ

* Адаптивность интерфейса

Использование кастомного хука use-mobile.tsx позволяет определить размер экрана и подстроить поведение компонентов под мобильные устройства. Это означает, что если приложение запущено на телефоне или планшете, элементы интерфейса будут менять своё расположение, размер и отступы для обеспечения оптимального взаимодействия. Можно представить это как перемещение офисной мебели под размеры комнаты – всё подстраивается под текущие условия.

* Уведомления и обратная связь

Хук use-toast.ts отвечает за показ всплывающих уведомлений. Сообщения об успешном расчёте, ошибках валидации или других системных событиях появляются в виде небольших всплывающих окон (тостов), которые информируют пользователя. Это помогает создать ощущение активного диалога между системой и пользователем, как если бы на ресепшене офиса появлялся сотрудник с важной информацией.

* Работа с изображениями

Папка public содержит изображения, используемые в проекте, например, placeholder-logo.png. Эти статические файлы используются для отображения логотипов банков или других графических элементов. Такой подход позволяет хранить все медиа-ресурсы отдельно и легко заменять или обновлять их, не затрагивая логику работы компонентов

# СЦЕНАРИИ РАБОТЫ И ПОЛЬЗОВАТЕЛЬСКИЙ ОПЫТ

* Начальный этап взаимодействия

При запуске приложения через команду npm run dev или команду electron запускается основной процесс, определённый в main.js. Окно Electron открывает веб-страницу, которая инициализирует приложение Next.js. Пользователь видит главную страницу с возможностью выбора ипотечных программ. Начальный экран отображает карточки банков с краткой информацией по программам: ставки, суммы, условия. Этот процесс можно сравнить с входом в шоурум, где на витринах расположены предложения о кредитовании.

* Ввод данных и пошаговый расчёт

После выбора карточки банка пользователь переходит к заполнению формы калькулятора (calculator-form.tsx). Каждое поле формы снабжено подсказками и валидацией, что позволяет избежать ошибок ввода. При нажатии кнопки «Рассчитать» данные отправляются на серверную часть, а затем результаты возвращаются в виде детализированного отчёта. Пользователь получает не только числовые значения, но и визуальное представление процесса погашения кредита через диаграммы.

* Обратная связь и уведомления

На каждом этапе приложение обеспечивает пользователя обратной связью. Если ввод неверен, появляются ошибки валидации. После успешного расчёта всплывают уведомления о выполненной операции. Это позволяет создать прозрачное взаимодействие между системой и клиентом – как если бы в офисе постоянно коммуницировали друг с другом через совещательные звонки и информационные панели.

# АЛГОРИТМЫ ВЫЧИСЛЕНИЯ И PAYLOAD ОБРАБОТКИ

* Математический аппарат расчётов

Основная часть вычислений происходит на сервере в файле route.py (папка api/calculate). Здесь реализованы алгоритмы для расчёта ежемесячных платежей, общей суммы выплат, а также распределения финансовых потоков во времени. В расчетах учитываются:

– Основная сумма кредита.

– Годовая процентная ставка.

– Срок кредита (в месяцах или годах).

– Дополнительные параметры, такие как наличие или отсутствие страховки, специальные условия для IT-специалистов и семей с детьми.

Аналогия здесь такова: если бы ипотека представляла собой сложный механизм мотора, то функции расчёта – это инженерный блок, где каждое звено (ставка, сумма, срок) влияет на итоговую работу двигателя. Каждый калькулятор получает «пайплайн» данных, обрабатывает его и возвращает результат, готовый для визуализации.

* Передача и получение данных

Между клиентской стороной (React-компоненты) и серверным модулем происходит обмен данными посредством API вызовов. JSON-пакеты, содержащие информацию о введённых пользователем значениях, отправляются на Python-сервер, где расчеты выполняются, а затем возвращаются в виде структурированного объекта MortgageResults. Это гарантирует согласованность форматов и позволяет легко интегрировать результаты в визуальные компоненты.

# РОЛЬ ТЕСТИРОВАНИЯ И ВЕРИФИКАЦИИ

Несмотря на то что в описании проекта не приводится набор модульных тестов, архитектура приложения заложена с учётом возможности написания тестов для каждого компонента. Разработчики могут:

– Писать тесты для валидации формы (calculator-form.tsx), проверяя, что при неверном вводе пользователь получает корректное сообщение об ошибке.

– Проверять корректность расчётов, вызываемых на сервере, сравнивая полученные результаты с ожидаемыми значениями.

– Тестировать компоненты визуализации (mortgage-charts.tsx), убеждаясь, что графики отображаются правильно в зависимости от входных данных.

Эта система тестирования помогает поддерживать качество приложения и быстро обнаруживать ошибки в логике расчёта или визуализации.

# ЭКОЛОГИЯ ПРОЕКТА И БУДУЩИЕ РАЗВИТИЯ

* Модульность и масштабируемость

Структура нашего проекта дает возможность легко расширять функционал за счёт добавления новых компонентов, хуков или API маршрутов. Например, можно добавить новую категорию ипотечных программ в banks.ts или расширить функциональность компонента калькулятора для поддержки дополнительных опций (рефинансирование, различные типы страховок и т.д.). Модульность позволяет разработчикам работать независимо над разными частями, что ускоряет процесс добавления новых возможностей. Возможности интеграции с Electron

Поддержка Electron позволяет запускать приложение не только как веб-приложение, но и как нативное настольное приложение для Windows (а в будущем – для macOS и Linux). Это значительно расширяет аудиторию проекта, поскольку пользователи, привычные к настольным приложениям, смогут пользоваться нашим калькулятором без обращения к веб-браузеру.

* Адаптация интерфейса и международная поддержка

Благодаря использованию гибких CSS-переменных, Tailwind CSS и современных React-компонентов, приложение легко адаптируется к различным языкам, размерам экрана и особенностям локальных рынков. Это позволяет в будущем выпускать версии калькулятора, адаптированные под требования разных стран: изменение формата валют, перевод текстов, настройка локальных параметров кредита.

# ЗАКЛЮЧЕНИЕ

В данном проекте была создана программа, которая полезна пользователям, без особых знаний, в планировании ипотеки. Так же, в заключении можно отметить, что наш проект – это комплексное и современное решение для расчёта ипотеки, объединяющее в себе богатый функционал, продуманную архитектуру и гибкость настройки. Каждый файл и каждая папка в проекте играют свою роль: от конфигурационных файлов, задающих правила работы приложения, до отдельных модулей визуализации, отвечающих за отображение итоговых данных. Систематизация кода посредством разделения на папки (app, components, data, hooks, lib, types) помогает поддерживать проект в чистоте и позволяет легко добавлять новые возможности.

Весь проект можно сравнить с большими офисными комплексами, где каждая «комната» отвечает за свою задачу. Пользовательский интерфейс – это витрина, которая дружелюбно встречает клиента, форма расчёта – это контрольная панель управления, а серверная логика – мощный двигатель, отвечающий за точность вычислений. Такой подход обеспечивает высокое качество, масштабируемость и удобство использования приложения, позволяя в будущем легко интегрировать новые модули и улучшения.

Наш проект включает настройку окружения для разработки на TypeScript и Next.js, поддержку современного CSS при помощи Tailwind и PostCSS, использование Electron для нативного запуска, а также модульную архитектуру, где каждый компонент, утилита или сервис имеют явное назначение и интегрированы в общий рабочий процесс. Все это способствует созданию высококачественного, отзывчивого и функционального калькулятора ипотеки, который может удовлетворить потребности самых требовательных пользователей.

Это подробное описание охватывает все аспекты проекта, от базовых конфигурационных файлов до тонкостей вычислительной логики и визуализации. Оно позволяет понять, как отдельные блоки проекта связаны между собой и какую роль играют в обеспечении целостного пользовательского опыта.

СПИСОК ЛИТЕРАТУРЫ

1. <https://java-course.ru/begin/interfaces/>
2. <https://foxminded.ua/ru/python-vse-komandy/>
3. <https://www.vtb.ru>
4. <https://www.pik.ru>
5. <https://alfabank.ru>
6. Поляков К. Ю.100 баллов по информатике. Решаем задачи ЕГЭ на языке Python : учебное пособие / К. Ю. Поляков.

2-е изд. - М. : Лаборатория знаний, 2025. - 367 с. : ил.

1. Поляков К. Ю. Информатика. Углубленный уровень: учебник для 11 класса : в 2 ч. Ч. 1 / К. О. Поляков, Е. А. Ере-мин. - 6-е изд., стереотиц." м. : БИНОМ. Лаборатория знаний, 2017. - 248 с. : ил.