**Приложение «Калорийный калькулятор»**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**1. Введение**

Приложение "Калорийный калькулятор" предназначено для пользователей, которые хотят отслеживать потребление калорий и контролировать свой рацион питания с целью достижения различных целей, таких как снижение веса, поддержание текущего веса или набор массы.

**2. Цели и задачи**

* Разработать интуитивно понятный интерфейс для ввода и отслеживания приема пищи.
* Обеспечить пользовательскую базу данными о продуктах с указанием их калорийности и питательных веществ.
* Внедрить функционал для расчета ежедневной нормы калорий в зависимости от целей пользователя.
* Реализовать систему напоминаний о приеме пищи и калибровке данных.
* Обеспечить возможность создания отчетов о потреблении калорий за день, неделю и месяц.

**3. Функциональные требования**

**3.1. Пользовательский интерфейс**

* Регистрация и авторизация пользователей.
* Главный экран с отображением текущих данных о потреблении калорий и рекомендованных значений.
* Экран для добавления продуктов с возможностью выбора из базы данных или ручного ввода.
* Экран отчетов с возможностью фильтрации по датам и типам пищи.

**3.2. База данных**

* Хранение информации о пользователях и их предпочтениях.
* База данных продуктов с информацией о калориях, макро- и микроэлементах.
* Функциональность для обновления и расширения базы данных продуктов.

**3.3. Расчет калорий**

* Алгоритм расчета суточной нормы калорий на основе введенных данных: возраст, пол, вес, рост, уровень активности и цели (снижение/поддержание/набор веса).
* Подсчет калорий в зависимости от введенных пользователем продуктов.

**3.4. Напоминания**

* Система уведомлений о приеме пищи и необходимости ввода данных.
* Возможность настройки частоты напоминаний.

**3.5. Отчеты**

* Генерация отчетов о потреблении калорий за выбранный период.
* Возможность визуализации данных (графики, диаграммы).

**4. Нефункциональные требования**

* Платформы: iOS, Android.
* Интерфейс: адаптивный и понятный, поддерживающий различные языки.
* Безопасность: защита личных данных пользователей.
* Производительность: быстрое реагирование на действия пользователей, минимальная задержка при загрузке данных.

**5. Технические условия**

* Языки программирования: Swift (iOS), Kotlin (Android).
* Серверная часть: Node.js с использованием Express для API.
* База данных: MongoDB или PostgreSQL.
* Использование внешних API для получения данных о продуктах (например, Nutritionix, Edamam).

**6. Бюджет и ресурсы**

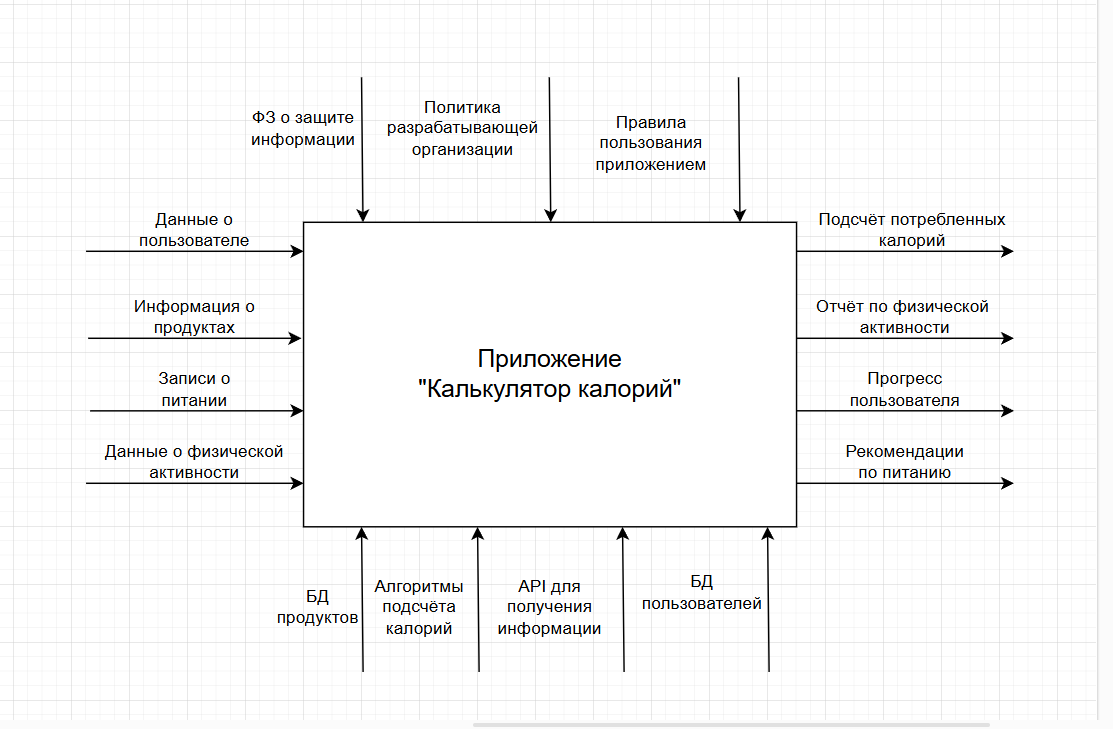
Бюджет и расходы на создание данного приложения могут варьироваться в зависимости от масштаба проекта, функциональных требований системы и используемых технологий. Бюджет заказчика – 500 тысяч рублей, ожидаемые расходы – 400 тысяч. Подробный расчет бюджета требует проведения анализа и составления бизнес-плана проекта.

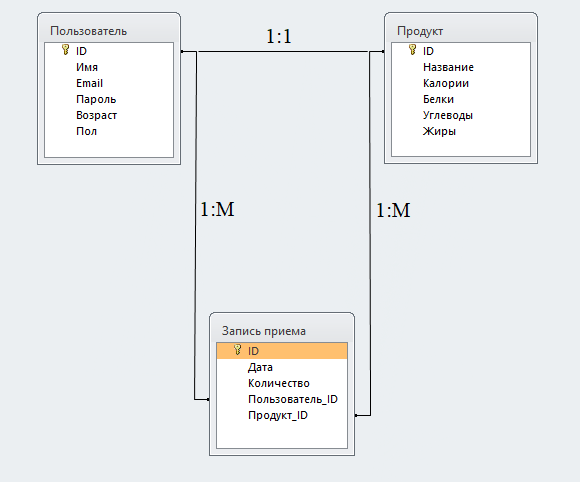
**7. Этапы разработки**

1. Сбор и анализ требований.
2. Проектирование архитектуры приложения.
3. Разработка пользовательского интерфейса.
4. Реализация и интеграция функциональности.
5. Тестирование приложения на наличие ошибок и багов.
6. Разработка пользовательской документации и справки.
7. Запуск бета-версии и сбор отзывов от пользователей.
8. Внедрение фидбека и доработка приложения.
9. Окончательный релиз и размещение в магазинах приложений.

**8. Сроки выполнения**

* Общая продолжительность разработки: 6 месяцев.
* Этап проектирования: 1 месяц.
* Этап разработки: 4 месяца.
* Этап тестирования и доработки: 1 месяц.

**IDEF0-ДИАГРАММА**

**ER-ДИАГРАММА**

**КАКИЕ ЯЗЫКИ И ТЕХНОЛОГИИ ИСПОЛЬЗУЮТСЯ В РАЗРАБОТКЕ И ПОЧЕМУ?**

При создании приложения "Калькулятор калорий" могут быть использованы следующие языки и технологии:

*1. Языки программирования:*

* Java/Kotlin (для Android): эти языки популярны для разработки приложений на платформе Android благодаря своей высокой производительности и возможностям.
* Swift (для iOS): Swift является основным языком для разработки приложений под iOS и обладает современным синтаксисом и безопасностью.
* JavaScript/TypeScript (для кросс-платформенных приложений): используются вместе с фреймворками, такими как React Native, для разработки приложений на нескольких платформах одновременно.

*2. Фреймворки и библиотеки:*

* React Native: Позволяет создавать кроссплатформенные мобильные приложения, что экономит время и ресурсы.
* Flutter: Также используется для кросс-платформенной разработки, обеспечивая высокую производительность и привлекательные UI.

*3. Базы данных:*

* SQLite: легковесная база данных, хорошо подходит для мобильных приложений, где важно хранить данные локально.
* Firebase: используется для облачного хранения данных и аутентификации пользователей.

*4. API и интеграции:*

* RESTful API: для взаимодействия с сервером и получения данных (например, о пище и калориях) было бы удобно использовать REST API.
* GraphQL: ориентирован на возможность получать только те данные, которые нужны, улучшая производительность.

*5. Инструменты для разработки:*

* Git: для контроля версий и совместной работы между разработчиками.
* Docker: позволяет создать изолированное окружение для разработки и тестирования приложения.

*Почему именно эти технологии:*

1. Популярность и поддержка сообщества: все упомянутые языки и фреймворки имеют большое сообщество, что упрощает решение проблем и получение помощи.
2. Кросс-платформенность: использование кросс-платформенных технологий помогает выйти на рынок быстрее и с меньшими затратами.
3. Высокая производительность: выбор языков и технологий, обеспечивающих высокую производительность, важен для создания отзывчивого приложения.

Эти языки и технологии помогут создать функциональное, эффективное и доступное приложение для пользователей.