

8 (911) 950-99-44 | esscale@mail.ru | esscale.ru

Прайс-лист на разработку программного обеспечения

Подход, основанный на прозрачной оценке сложности

Здравствуйте!

Мы верим, что ключ к успешному проекту — это честность и предсказуемость. Вместо того чтобы пытаться "угадать" точное количество часов, мы используем современный подход из Agile-разработки — оценку по относительной сложности.

Это позволяет нам дать точный прогноз бюджета и сроков, основываясь не на интуиции, а на декомпозиции проекта и командной оценке.

Наша философия ценообразования: Esscale Points

Мы оцениваем каждую задачу в проекте в **Esscale Points (EP)** — условных единицах, отражающих её **объем, сложность и риски**. Для этого мы используем шкалу, основанную на числах Фибоначчи (1, 2, 3, 5, 8, 13...).

Почему это лучше для вас?

• Прозрачность: Вы видите, какие задачи являются самыми трудоемкими и почему.

- Точность: Оценивать задачи относительно друг друга («эта задача в два раза сложнее той») намного точнее, чем гадать в абсолютных часах.
- Гибкость: Если в проект нужно добавить новую фичу, мы легко можем оценить её в Esscale Points и показать, как это повлияет на общий бюджет и сроки.

$$Total\,cost = \sum_{i=1}^k EP_i^* \,Price\,of\,ep$$

$$EP_i^{} = \prod_{k=1}^{n_i} Coeff_{ik}^* \,^*f_k^{}(i) \sum_{j=1}^{m_i} ep_{ji}^* \,^*g_j^{}(i) \;\;\text{,}$$

где

$$g_{j}(i) = \left\{ egin{array}{ll} 1,$$
 если услуга выполняется $\end{array}
ight.$

$$f_{k}(i) = \begin{cases} 1, ext{ecли услуга выполняется} \\ rac{1}{ ext{Coef} f_{ik}}, ext{ если услуга не осуществляется} \end{cases}$$

Где EP_i - это сумма значений за один і-раздел;

 $\overline{Price\ of\ ep}$ – это фиксированная цена за \overline{ep} , которая зависит от состава команды, работающей над проектом;

 $Coeff_{ik}^{-}$ это коэффициентное значение услуги (при необходимости только в Backend) в i-разделе, которая в той или иной степени усложняет работу и которую трудно оценить по числовой шкале Фибоначчи;

 $ep_{_{ji}}$ - это значение оценки услуги по предложенному прайсу.

Ориентировочная оценка проектов в Esscale Points

Мы разделили проекты на три категории по объему и сложности:

Проект "Бессерверное приложение"

Мобильное приложение не требующее написания собственного сервера.

- Объем: до 200 Esscale Points
- Данный проект может включать:
 - ∘ Проектирование и UI/UX дизайн
 - Реализация анимаций и переходов
 - Сложная логика и математические модели
 - Применение сложных алгоритмов
 - Работа с медиаконтентом
 - ∘ Тестирование CI/CD

Проект "Стандарт"

Полноценное приложение с серверной частью.

- **Объем:** от 250 до 340 Esscale Points
- Данный проект может включать (все из Бессерверного приложения +):
 - Регистрация/авторизация, личный кабинет
 - o Push-уведомления
 - о Серверная логика и аналитика
 - Внедрение сторонних АРІ
 - о Поддержание банковских транзакций
 - о Простые чаты
 - ∘ Поддержание ~10000 RPS
 - ∘ Обработка статики

Проект "Pro / Комплексный"

Сложный продукт с нестандартной логикой, высокой нагрузкой или множеством интеграций.

- Объем: от 340 Esscale Points
- Данный проект может включать (все из Стандарт +):
 - о Сложная backend архитектура
 - Масштабирование
 - ∘ Поддержание ~1000000 RPS
 - Сложная аналитика
 - Разработка собственных математических моделей
 - ∘ Интеграция или разработка ML и DL
 - Работа с медиаконтентом в реальном времени

Детализированный каталог услуг и оценка в Esscale Points

1. Backend Разработка (C++, Boost)

Nº	Услуга	Краткое описание	0ценка (EP)	Обоснование оценки
	Ар	хитектура и Бизнес-	логика	
1.1	Проектировани е масштабируемо й и отказоустойчи вой архитектуры	Создание фундамента сервиса, способного выдерживать высокие нагрузки, работать без сбоев и легко расширяться в будущем.	8-13	Сложность: Требует глубокого опыта и стратегического мышления для предвидения узких мест. Риски: Ошибки на этом этапе наиболее дороги в исправлении.
1.2	Реализация основной бизнес-логики сервера	Написание ключевого кода, который решает задачи вашего	8-13	Объем: Обычно это самая большая часть кодовой базы.

		бизнеса (обработка заказов, управление пользователями, расчеты и т.д.).		Сложность: Требует полного погружения в бизнес-процессы заказчика и их точной реализации в коде.
1.3.1	Реализация многопоточнос ти сервера (коэффициент) Реализация асинхронности сервера (коэффициент)	Внедрение механизмов для одновременной обработки тысяч запросов без блокировок, используя современные подходы C++.	1,3	Сложность: Высокая когнитивная нагрузка. Риски:Потенциал ьные трудноуловимые ошибки (гонки данных, deadlock), которые критически влияют на стабильность.
1.4	Рефакторинг существующего кода (коэффициент)	Модернизация кода для повышения его надежности, скорости работы, соответствия стандартам и упрощения дальнейшей поддержки.	1,5-3	Объем: Часто затрагивает большие участки кодовой базы. Риски:Необходим о глубоко понимать существующий код и не сломать рабочую логику.

1.5	Выбор и обоснование структур данных и алгоритмов	Подбор и реализация наиболее эффективных решений для высокой производительнос ти и экономии ресурсов (память, CPU).	3	Сложность: Требует сильных теоретических знаний и умения применять их на практике для решения конкретных задач.
1.6	Внедрение API сторонних сервисов	Интеграция с внешними системами (платежные шлюзы, службы доставки, социальные сети) для расширения функционала.	5-8	Сложность: Зависит от качества документации внешнего API. Риски: Нестабильная работа внешнего сервиса может влиять на ваш продукт.
1.7	Обработка банковских транзакций	Реализация логики взаимодействия с платежными системами, обеспечение безопасности и атомарности операций.	8	Риски: Максимальная ответственность . Любая ошибка может привести к финансовым потерям. Требует строгого следования протоколам безопасности.

1.8	Обработка электронных подписей	Интеграция с криптографически ми сервисами для верификации и создания электронных подписей.	8	Сложность: Требует знаний в области криптографии и стандартов безопасности. Высокие требования к надежности.
		Сетевое взаимодейст	твие	
2.1	Обработка и хранение сетевых подключений по HTTP (Beast)	Реализация логики приема, обработки и ответа на стандартные веб-запросы.	3	Объем: Стандартная, но объемная задача, являющаяся основой любого веб-сервиса. Требует аккуратной работы с ресурсами.
2.2	Обработка и хранение сетевых подключений по WebSocket	Реализация постоянных двусторонних соединений для real-time функций (чаты, уведомления, биржевые данные).	5	Сложность: Управлять жизненным циклом множества постоянных соединений сложнее, чем обработка stateless HTTP-запросов.

2.3	Обработка и парсинг JSON формата	Написание кода для чтения, валидации и преобразования данных в формате JSON, используемого в большинстве API.	3	Объем: Рутинная, но необходимая задача. Сложность возрастает при работе со сложными вложенными структурами и валидацией.
2.4.1	Обработка бинарных фреймов	Реализация работы с бинарными протоколами для высокопроизводит ельного обмена данными (игры, стриминг).	5	Сложность: Требует точной работы с байтами, битами и структурами данных. Отладка сложнее, чем с текстовыми форматами.
2.4.2	Обработка текстовых фреймов	Реализация работы с текстовыми протоколами.	3	Сложность: Верный парсинг формата строк. Обработка строковых данных.
2.5	Обработка статических и медиа файлов	Реализация отдачи файлов (изображения, документы, видео) с сервера, включая	5-8	Объем: Включает в себя настройку МІМЕ-типов, кэширования, потенциально —

	Pa	оптимизацию и контроль доступа. юбота с базами даннь	ıx (БД)	работу с большими файлами.
3.1	Подключение к БД и логика переподключен ия	Настройка соединения с базой данных и реализация надежной логики автоматического восстановления связи при сбоях.	5	Риски: Нестабильное соединение с БД может парализовать весь сервис. Требует тщательного тестирования.
3.2	Написание структуры данных БД (схемы, таблицы)	Проектирование логической структуры базы данных, определение таблиц, полей и связей между ними.	5	Сложность: Ошибки в проектировании схемы приводят к проблемам с производительно стью и сложностью поддержки в будущем.
3.3	Написание процедур и триггеров БД	Создание хранимых процедур и триггеров для реализации сложной логики на стороне самой базы данных.	5	Сложность: Перенос логики в БД усложняет отладку и тестирование. Требуются глубокие знания конкретной СУБД.

3.4	Оптимизация запросов и структуры БД	Анализ медленных запросов, добавление индексов, нормализация/ден ормализация таблиц для ускорения работы.	5	Сложность: Требует навыков профилирования и глубокого понимания того, как работает планировщик запросов СУБД.
3.5	Реализация системы репликации БД	Настройка синхронизации данных между несколькими серверами БД для отказоустойчивос ти и распределения нагрузки.	5	Сложность: Конфигурация и поддержка репликации — сложная инфраструктурна я задача. Риски: Неправильная настройка может привести к потере данных.
		Кэширование (Redi	.s)	
4.1	Кэширование запросов к БД	Внедрение Redis для сохранения результатов частых и тяжелых запросов к БД, что снижает нагрузку на нее.	5	Объем: Стандартная практика, требующая аккуратной реализации логики инвалидации кэша.

4.2	Кэширование	Использование	5	Сложность:
	общей	Redis для		Необходимо
	информации	хранения сессий,		правильно
	для	конфигураций и		· спроектировать
	масштабирован	других общих		КЛЮЧИ И
	ия	данных,		структуры
		доступных всем		данных в Redis
		экземплярам		для
		сервиса.		эффективного
				использования.
4.3	Реализация	Использование	8	Сложность:
	подписок	механизма		Требует
	Redis	Publish/Subscrib		проектирования
	(Pub/Sub)	е для обмена		асинхронной
		сообщениями		архитектуры
		между		обработки
		компонентами		сообщений.
		системы в		
		реальном		
		времени.		
		Безопасность		
5.1	Система	Полный цикл: от	3-8	Объем: Включает
	авторизации и	регистрации с		работу с БД,
	регистрации	подтверждением		хэширование
	пользователей	по почте до		паролей,
		входа в систему		генерацию
		и управления		токенов.
		сессиями.		Риски:
				Уязвимости в
				этом модуле
				критичны для
				безопасности.

5.2	Хэширование и безопасное хранение паролей	Использование современных и надежных криптографически х алгоритмов (например, bcrypt, Argon2) для защиты паролей.	1	Сложность: Хотя сама реализация проста с готовыми библиотеками, выбор правильного алгоритма и параметров требует знаний.
5.3	Настройка SSL шифрования	Обеспечение шифрования всего трафика между клиентом и сервером для защиты данных от перехвата.	3	Объем: Включает генерацию сертификатов (например, через Let's Encrypt) и настройку веб-сервера.
5.4	Защита от SQL-инъекций	Применение параметризованны х запросов и санитизации всех входящих данных для предотвращения атак на БД.	3	Риски: Обязательная мера безопасности. Оценка невысока, т.к. это должно быть стандартом, а не опцией.
5.5	Защита от брутфорс-атак	Внедрение механизмов ограничения количества попыток входа,	5	Сложность: Требует реализации логики отслеживания и

		капчи или временной блокировки IP.		блокировки подозрительной активности.
5.6	Защита от Thundering Herd	Применение техник на уровне ОС или приложения для предотвращения "эффекта толпы" при пробуждении множества процессов.	5	Сложность: Низкоуровневая и специфическая задача, требующая глубокого понимания работы ОС и сетевого стека.

2. DevOps и Инфраструктура

Nº	Услуга	Краткое описание	Оценка (EP)	Обоснование оценки
		Сборка и Контейнери	зация	
1.1	Линковка зависимостей в CMake	Написание сборочных скриптов для корректного подключения всех необходимых библиотек и компиляции проекта.	1-3	Сложность: CMake имеет сложный синтаксис. Неправильная линковка приводит к ошибкам компиляции или выполнения.
1.2	Настройка Docker	Создание Dockerfile и docker-compose.ym l для	3-5	Объем: Обеспечивает идентичность окружения на

		контейнеризации		всех машинах.
		приложения и его		Требует
		зависимостей (БД,		настройки сети,
		Redis).		томов и
				переменных
				окружения.
				- Спружении
		Веб-сервер (Ngin	x)	
2.1	Настройка	Конфигурация	3	Объем:
	проксирования	Nginx как		Стандартная, но
	Nginx	обратного прокси		обязательная
		для		задача для
		перенаправления		любого
		запросов на ваше		веб-сервиса.
		С++ приложение.		
2.2	 Настройка	 Конфигурация	2	Объем: Простая
	раздачи	Nginx для		' конфигурационная
	· ·· статики Nginx	эффективной		задача, которая
		отдачи		дает заметный
		статических		 прирост
		файлов (CSS, JS,		 производительнос
		картинки),		ти.
		разгружая		
		основное		
		приложение.		
			_	
2.3	Конфигурация	Настройка Nginx	5	Сложность:
	балансировщик	или другого		Требует
	а нагрузки	балансировщика		понимания
		для распределения		алгоритмов
		трафика между		балансировки и
		несколькими		настройки health
		экземплярами		checks для
		приложения.		обеспечения

				отказоустойчивос			
	Развертывание Redis						
3.1	Развертывание единственного экземпляра Redis	Установка и базовая настройка одного сервера Redis.	2	Объем: Простая задача, подходит для разработки и небольших проектов.			
3.2	Pазвертывание Redis HA / Sentinel	Настройка кластера из нескольких серверов Redis с автоматическим переключением на резервный узел при сбое основного.	5	Сложность: Требует настройки 3+ серверов Sentinel и понимания кворума. Риски: Неправильная настройка может привести к split-brain.			
3.3	Развертывание Redis Cluster	Настройка шардированного кластера Redis для горизонтального масштабирования данных и производительност и.	8	Сложность: Значительно сложнее, чем Sentinel. Требует планирования топологии, управления слотами и понимания работы gossip-протокола			

3. Frontend Разработка (Web: React / Vue / Angular, JavaScript/TypeScript)

Nº	Услуга	Краткое описание	Оценка (EP)	Обоснование оценки
	Пр	ооектирование и Архи	тектура	
1.1	Анализ UI/UX требований и макетов (Figma, Adobe XD)	Изучение дизайна и технического задания для планирования разработки, выявления сложных моментов и вопросов.	3	Сложность: Требует внимательности и хорошей коммуникации с дизайнером/мене джером. Основа для всей последующей работы.
1.2	Проектирование компонентной архитектуры	Разделение интерфейса на логические, переиспользуемые компоненты (кнопки, формы, карточки) для чистоты кода и скорости разработки.	8	Сложность: Крайне важный этап. Ошибки в архитектуре приводят к сложностям в поддержке и масштабировании проекта.
1.3	Выбор и настройка архитектуры	Выбор и внедрение глобального хранилища данных	8	Сложность: Ключевое архитектурное решение.

	управления состоянием	(Redux, MobX, Vuex) для управления состоянием приложения.		Неправильный выбор может привести к проблемам с производительно стью и излишней сложности кода.
	E	Верстка и UI/UX Реал	изация	
2.1	Верстка экранов и компонентов по макетам	Преобразование статичных дизайн-макетов в интерактивные веб-страницы с использованием HTML, CSS и JS/TS.	5	Объем: Основной объем "видимой" работы. Сложность зависит от точности следования макету (pixel-perfect)
2.2	Реализация адаптивного и отзывчивого дизайна	Обеспечение корректного отображения и функционирования интерфейса на всех устройствах: от смартфонов до широкоформатных мониторов.	8	Сложность: Требует глубоких знаний CSS (Flexbox, Grid, Media Queries) и тщательного тестирования на множестве разрешений.
2.3	Реализация сложных кастомных UI-элементов и анимаций	Создание нестандартных элементов (графики, дашборды,	13	Сложность: Часто требует написания сложной логики на

2.4	Реализация тем оформления (светлая/темна я тема)	drag-and-drop) и плавных анимаций для улучшения пользовательског о опыта. Внедрение механизма переключения цветовых схем во всем приложении.	5	ЈаvaScript/Туре Script или освоения специализирован ных библиотек. Объем: Требует проработки всей палитры цветов и обеспечения того, чтобы все компоненты поддерживали смену темы.
	Логи	ка и Взаимодействие	c Backend	
3.1	Реализация сложной клиентской логики	Написание кода для обработки действий пользователя: валидация форм, фильтрация/сорти ровка данных, расчеты в реальном времени.	13	Объем: Это "мозг" фронтенд-прилож ения. Объем кода напрямую зависит от сложности бизнес-требован ий.
3.2	Интеграция с API бэкенда (HTTP-запросы)	Настройка обмена данными с сервером: отправка запросов, обработка ответов, управление	8	Объем: Стандартная, но объемная задача, включающая в себя работу с асинхронностью и

		состоянием загрузки и ошибками.		преобразованием данных.
3.3	Реализация аутентификации и авторизации на клиенте	Создание форм входа/регистраци и, работа с токенами (JWT), управление доступом к разным разделам приложения.	8	Риски: Критически важная для безопасности задача. Требует аккуратной работы с хранением токенов для предотвращения XSS-атак.
3.4	Peaлизация real-time функций (WebSocket)	Подключение к WebSocket для получения обновлений от сервера в реальном времени (чаты, уведомления, графики).	8	Сложность: Управление состоянием, которое может измениться в любой момент по инициативе сервера, сложнее традиционного "запрос-ответ".
	C	Оптимизация и Тестир	ование	
4.1	Оптимизация производительн ости загрузки (Code Splitting, Lazy Load)	Настройка сборки проекта для уменьшения начального размера бандла и ускорения	13	Сложность: Требует глубоких знаний инструментов сборки (Webpack, Vite)

	загрузки приложени	1Я.	и анализа производительно сти. Оказывает огромное влияние на UX.
4.2 Написани и Integr тестов		для ой гов и огики, а ащения	Объем: Написание тестов может занимать значительное время, но критически важно для долгосрочной поддержки больших проектов.

4. Frontend Разработка (Mobile: Flutter, Dart)

Nº	Услуга	Краткое описание	Оценка (EP)	Обоснование оценки			
	Проектирование и Архитектура						
1.1	Проектирование структуры виджетов и состояния приложения	Разработка архитектуры экранов и компонентов (виджетов),	8	Сложность: Фундамент приложения. Правильная иерархия			

		определение потоков данных и состояний в приложении.		виджетов и управление состоянием напрямую влияют на производительнос ть и простоту поддержки.
1.2	Выбор и обоснование архитектуры состояния (Provider, BLoC, Riverpod)	Выбор глобальной стратегии управления состоянием, которая будет использоваться во всем приложении.	8	Сложность: Ключевое решение, которое сложно изменить в будущем. Требует опыта для выбора оптимального подхода под конкретный проект.
1.3	Разработка схемы навигации между экранами	Проектирование и реализация переходов между экранами, передача параметров, обработка deep linking.	5	Сложность: Может стать нетривиальной задачей в больших приложениях с вложенной навигацией и сложными сценариями.
	В	ерстка и UI/UX Реал	изация	
2.1	Верстка экранов и	Создание интерфейса	5	Объем: Основная работа по

	компонентов по	приложения		созданию
	макетам с	путем		визуальной части
	использованием	композиции		приложения.
	Flutter	виджетов в		
	Widgets	соответствии с		
		дизайн-макетами		
2.2	Планирование и	Обеспечение	8	Сложность:
	реализация	корректного		Требует
	адаптивного	отображения		использования
	дизайна	интерфейса на		специальных
		смартфонах и		виджетов
		планшетах с		(LayoutBuilder,
		разным размером		MediaQuery) и
		экрана и		продумывания
		ориентацией.		логики для
				разных
				конфигураций
				экрана.
				экрапа.
2.3	Реализация	Создание	5	Сложность:
	сложных	кастомных,		Flutter
	анимаций и	плавных		предоставляет
	переходов	анимаций для		мощные
		элементов		инструменты для
		интерфейса и		анимации, но их
		переходов между		освоение и
		экранами.		применение для
		опранами.		создания
				сложного UX
				требует времени.
		Логика и Интеграц	ция	
3.1	Реализация	Написание	13	Сложность:
	сложной	бизнес-логики		Логика

	клиентской логики (валидация, кэширование, оффлайн-режим)	приложения. Реализация оффлайн-режима добавляет значительную сложность в синхронизации данных.		оффлайн-доступа требует проектирования локальной базы данных (например, SQLite, Hive) и стратегии синхронизации с сервером.
3.2	Вызов АРІ бэкенда (http, dio), обработка ответов	Интеграция с сервером для получения и отправки данных. Включает сериализацию/де сериализацию JSON.	ω	Объем: Стандартная, но критически важная задача, требующая обработки множества сценариев (успех, ошибка, отсутствие сети).
3.3	Интеграция с нативными функциями платформы (камера, геолокация, сенсоры)	Использование плагинов для доступа к аппаратному обеспечению устройства (камера, GPS, акселерометр, Bluetooth).	13	Сложность: Требует работы с platform channels, обработки запросов на разрешения от пользователя и учета особенностей iOS и Android. Риски: Высокий

				риск
				platform-specifi
				с багов.
	Te	стирование и Развер	тывание	
4.1	Ручное и	Проверка работы	8	Объем: Очень
	автоматизирова	приложения на		трудоемкий
	нное	реальных		процесс,
	тестирование	устройствах и		необходимый для
	UI на разных	эмуляторах для		обеспечения
	устройствах и	выявления		качества.
	эмуляторах	багов,		Автоматизация
		специфичных для		(Widget Tests,
		платформ или		Integration
		моделей.		Tests) снижает
				ручной труд в
				будущем.
4.2	Сборка и	Подготовка	8	Сложность:
	публикация	сборок		Процесс связан с
	приложения в	приложения,		бюрократическими
	App Store и	оформление		требованиями
	Google Play	страниц в		платформ,
		сторах,		управлением
		прохождение		сертификатами и
		ревью и		профилями, что
		публикация.		часто вызывает
				непредвиденные
				сложности.

Проект: Рейтинговое приложение

Задача: Расчет КРІ, возможность финансового управления предприятия через приложение, предоставление основной отчетности по деятельности.

1. Оценка задач в Esscale Points (EP):

Написание сервера без обработки изображений:

1.1. Backend Разработка (C++, Boost):

- Проектирование масштабируемой и отказоустойчивой архитектуры: 8 ЕР
- Реализация основной бизнес-логики сервера: 13 ЕР
- Реализация многопоточности сервера (коэффициент): 1,3
- Реализация асинхронности сервера (коэффициент): 1,3
- Выбор и обоснование структур данных и алгоритмов: 3 ЕР
- Обработка и хранение сетевых подключений по WebSocket: 5 EP
- Обработка и парсинг JSON формата: 3 EP
- Обработка текстовых фреймов: 3 ЕР
- Подключение к БД и логика переподключения: 5 ЕР
- Написание структуры данных БД: 5 ЕР
- Написание процедур и триггеров БД: 5 ЕР
- Оптимизация запросов и структуры БД: 5 ЕР
- Реализация системы репликации БД: 5 ЕР
- Кэширование запросов к БД: 5 ЕР
- Кэширование общей информации для масштабирования: 5 ЕР
- Реализация подписок Redis: 8 EP
- Система авторизации и регистрации пользователей: 8 ЕР
- Хэширование и безопасное хранение паролей: 1 ЕР
- Настройка SSL шифрования: 3 EP
- Защита от SQL-инъекций: 3 EP
- Защита от брутфорс-атак: 5 ЕР
- Защита от Thundering Herd: 8 EP

1.2. DevOps и Инфраструктура

- Линковка зависимостей в CMake: 3 EP
- Настройка Docker: 5 EP
- Настройка проксирования Nginx: 3 EP
- Развертывание единственного экземпляра Redis: 2 EP

Добавление обработки изображения:

1.3. Backend Разработка (C++, Boost):

- Обработка и хранение сетевых подключений по HTTP (Beast): 3 EP
- Обработка бинарных фреймов: 5 ЕР
- Обработка статических и медиа файлов: 8 ЕР

1.4. DevOps и Инфраструктура

• Настройка раздачи статики Nginx: 2 EP

Итого по Backend-разработке: 221,18 Esscale Points

1.5. Frontend Разработка (Mobile: Flutter, Dart)

- Проектирование структуры виджетов и состояния приложения: 8 ЕР
- Выбор и обоснование архитектуры состояния: 8 ЕР
- Разработка схемы навигации между экранами: 5 ЕР
- Верстка экранов и компонентов по макетам с использованием Flutter Widgets: 5 EP
- Планирование и реализация адаптивного дизайна: 8 ЕР
- Реализация сложных анимаций и переходов: 5 ЕР
- Реализация сложной клиентской логики: 13 ЕР
- Вызов АРІ бэкенда (http, dio), обработка ответов: 8 EP
- Ручное и автоматизированное тестирование UI на разных устройствах и эмуляторах: 8 EP
- Сборка и публикация приложения в App Store и Google Play: 8 EP

Итого по Frontend-разработке: 76 Esscale Points

Итого Esscale Points: 297,18 Esscale Points

2. Расчет стоимости:

- Предположим, стоимость 1 Esscale Point для вашего проекта составляет 4 150 ₽.
- Итоговая стоимость проекта: 297,18 EP × 4 150 ₽/EP = 1 233 297 ₽

3. Расчет сроков:

- Наша средняя производительность (velocity) около **15 EP** в неделю.
- Примерный срок выполнения: 297,18 EP / 15 EP в неделю \approx 20 недель.

Следующие шаги

Готовы перевести идею в конкретный план? Свяжитесь с нами для бесплатной консультации. Мы:

- 1. Обсудим цели и задачи вашего проекта.
- 2. Проведем первичную декомпозицию и оценку объема в Esscale Points.
- 3. Подготовим для вас детализированное коммерческое предложение.

Давайте создадим продукт, которым будем гордиться!