

Создание проекта включало две ключевые составляющие: реализацию логики калькулятора и организацию API с помощью HTTP и gRPC.

Основная сложность реализации логики заключалась в оптимизации производительности при условии, что каждая арифметическая операция искусственно замедлена на 50 мс. Первоначальная попытка группировки независимых операций для конкурентного выполнения не дала эффекта. В итоге была использована комбинация `sync.Map` (Обычный `map` приводил к ошибкам из-за одновременной записи при большом количестве переменных) и `WaitGroup` для реализации механизма ожидания зависимых переменных. Например, если переменная `x` зависит от ещё не вычисленной `y`, соответствующая горутина ждет готовности `y`. Такой подход позволил достичь оптимальной скорости работы — программа обрабатывает 10 000 независимых инструкций за ~100 мс.

Для обеспечения модульности и тестируемости был применён паттерн "Сервис". Структура `Calculator` инкапсулирует состояние переменных (хранящееся в `sync.Map`) и всю вычислительную логику.

Реализация HTTP-интерфейса на базе стандартного пакета `net/http` была простой. Для gRPC потребовалось:

1. Описать API в `.proto`-файле
2. Сгенерировать серверный и клиентский код
3. Реализовать серверную логику, адаптирующую gRPC-вызовы к `Calculator`

Swagger-документация генерировалась автоматически на основе аннотаций в `main.go`, что упростило документирование.

Все пакеты были разделены логически в структуре проекта, написаны юнит тесты для `calc.go`, для `main.go` также были написаны тесты для проверки работы нод `http` и `grpc`, но они скорее интеграционные.