По итогам первой части нашего курса вы умеете:

1. Разбираться в простых конструкциях языка, понимать его синтаксис и структуру. Знаете как объявляются переменные, какие типы данных в GO бывают. Умеете объявлять и присваивать их, умеете делать циклы и ветвления.
2. Писать простые приложения на GO. Решать вычислительные и логические задачи с помощью этого языка.
3. Работаете со слайсами, отличаете их от массивов. А также прикоснулись к более сложным типам: строки, хеш таблицы и умеете создавать свои структуры для решения задач.
4. Умеете работать с многопоточностью и горутинами. Разбираетесь в синхронизациях горутин, применяете waitgroups и каналы, можете написать приложение, использующее параллельность и успешно завершить его с правильными результатами.
5. Разбираетесь в базовых строительных блоках прикладных приложений
    a. Умеете писать и читать файлы, безопасно выходить из приложений и закрывать файлы. Можете делать весь спектр операции с ними.
    b. Понимаете разницу между форматами обмена данных и применяете их, можете написать свой клиент, для забора данных в основных популярных форматах откуда угодно, а также подключаться на более низких уровнях.
    c. Понимаете, как структурировать подключение к БД, для чего нужны различные драйвера и в чем разница между популярными типами БД.
    d. Можете написать свой HTTP сервер, правильно разложить его структуру, для использования и расширения в дальнейшем, подключить к базе данных и написать любое backend веб приложение.

Для закрепления всего пройденного материала мы выполняли различные задачи с алгоритмами, и самое важное, у нас должен был получиться итоговый сервис на Go.

В этом сервисе мы разработали самый важный для функционала элемент - Checker, который умеет осуществлять проверки сторонних сервисов на предмет health и собирать метрики.

Мы начинали писать код для сервиса используя базовые типы данных, по мере углубления мы реализовывали интерфейсы и композицию. Для поведения описывали методы структур. Научили наш чекер выполнять проверки в фоновом режиме и синхронизироваться сигналами с “внешним” миром (например останавливаться или ждать завершения фоновых проверок). На примере этого же сервиса мы с вами познакомились с концепцией таймаутов, которые можно применять как к методам внутри проекта, так и к вызову других сервисов по сети.