Коллеги, вы меня огорчаете

В июле и августе 2020 года я, с подачи Григория Петрова, проводил для компании Evrone технические интервью на позицию Senior Golang Backend developer. И, видимо, буду вынужден продолжать проводить, о чём ниже.  
  
Задача формулировалась как «найти человека, который сможет задать и поддерживать высокий уровень профессионализма в применении языка Go». То есть, сформулирована она была по-человечески, перевод на канцелярит — мой. Под эту задачу я сформировал новый опросник вместо того, которым пользовался несколько лет — старый был с жестким закосом под DevOps. Методику, которой я пользуюсь для создания опросников и количественной оценки соответствия кандидатов, я излагал в своем докладе «Техническое интервью как инженерная задача» на конференции Saint TeamLead 2019.  
  
И вот что я хочу сказать вам, коллеги: вы меня огорчаете.  
  
 Благодаря усилиям хэдхантеров Evrone Екатерины Тхоржевской и Анны Кудряшовой — спасибо им! — до меня добираются только резюме интересных кандидатов. Я их читаю и вижу вполне релевантный, а зачастую интересный профессиональный путь. Сложные и интересные проекты, ответственные должности. Слушая обязательный «Расскажите о себе» этап интервью, я вижу упорную и плодотворную работу. Прям бери и нанимай!  
  
Но когда дело доходит до технической части и опросника, перспективы найма начинают выглядеть уже не так радужно. Кисло они начинают выглядеть, честно говоря! Смотрите: средний балл по опроснику — 3.3 по шкале 0-9. 3.3, Карл! 3.3 — это, с моей точки зрения, уровень junior. Middle должен набирать, я считаю, 5+. Senior — 8+.  
  
Почему так? Почему опытные, квалифицированные разработчики набирают такой низкий балл? У меня есть, конечно, предположение, и я им поделюсь в конце статьи. А пока вернемся к тому, о чем я хочу с вами поговорить — к моему огорчению. Эта статья имеет своей целью ни много ни мало, — исправить ситуацию. Я хочу пройтись по своему опроснику и рассказать, какие именно неправильные ответы мне приходится выслушивать, а после этого задать вам наводящие вопросы, поиск ответов на которые поможет вам в следующий раз получить на моем интервью 9.  
  
Нет, я не дам вам правильных ответов. Во-первых, потому, что правильных ответов на большинство этих вопросов не существует. Во-вторых, цель моего интервью — оценить глубину системы знаний кандидата, а не его умение выдать ответы, которые понравятся интервьюеру. И я уверен — знания, полученные в процессе поиска ответов на наводящие вопросы, позволят вам продвинуться, как профессионалам и гоферам. Иначе я не стал бы писать эту статью, пережил бы свое огорчение тихо в уголке.  
  
Итак, опросник:  
  
1. Go — императивный или декларативный? А в чем разница?  
  
  
Что я хочу оценить: знакомство с разными подходами к реализации бизнес-логики.  
Самый популярный неправильный ответ: «Я не силен в теории». Очень жаль, что ты не силен в этой теории, %USERNAME%. Если бы ты был в ней силен, ты бы знал, почему некоторые вещи на Go получаются очень легко и хорошо, а некоторые надо прям вымучивать.  
Наводящие вопросы: SQL — императивный или декларативный? А Dockerfile? А файл настройки github actions?  
  
2. Что такое type switch?  
  
  
Что я хочу оценить: знакомство с нашим (скудным) инструментарием работы с системой типов в runtime.  
Самый популярный неправильный ответ: «Я не знаю». Очень странно — информация о type switch есть даже в Go tour.  
Наводящие вопросы: как реализовать в Go тип-сумму, который может содержать в себе значения int64|float64|complex? Как реализовать для такого типа метод Add(int64)?  
  
3. Как сообщить компилятору, что наш тип реализует интерфейс?  
  
  
Что я хочу оценить: хорошо ли кандидат понимает, на чем основана концепция интерфейсов в Go.  
Самый популярный неправильный ответ: «Я не знаю». Справедливости ради, именно этот вопрос редко вызывает затруднения. А когда вызывает, сам кандидат весьма этим удивлен: «Я же миллион интерфейсов написал. Как компилятор понимает, что именно я реализовал?...» Ну — он умный, компилятор. И внимательный.  
Наводящие вопросы: что такое duck typing? К чему он применяется в Go?  
  
4. Как работает append?  
  
  
Что я хочу оценить: знаком ли кандидат с базовыми концепциями управления памятью в Go. Самыми базовыми.  
Самый популярный неправильный ответ: «Он увеличивает capacity». Если продолжать настойчиво спрашивать: «Как он это делает?», — кандидат довольно быстро приходит к правильному ответу. Видимо, эти подробности настолько шокирующие, что забыть их трудно.  
Наводящие вопросы: как бы вы реализовали разреженный массив в Go? А без использования map?  
  
5. Какое у slice zero value? Какие операции над ним возможны?  
  
  
Что я хочу оценить: помнит ли кандидат, что вообще можно делать со слайсом, и как ведут себя операции на граничных значениях. Почему это важно? «Почему это важно?» — был бы отличный вопрос для интервью, если бы я придумал, как его корректно задавать.  
Самый популярный неправильный ответ: «Можно делать len ()… и cap ()… наверное…» Операций со слайсами существенно меньше 10. И мы все — 100% — применяем их в своей повседневной работе. Надо просто пересчитать их в уме...  
Наводящие вопросы: каков будет результат append([]string(nil), "")? А append([]string(nil), []string(nil)...)? А почему? А range append([]string(nil), []string(nil)...) как отработает?  
  
6. Как устроен тип map?  
  
  
Что я хочу оценить: насколько интересно кандидату, как именно ложатся в память наши байтики. Map, возможно, самая важная из стандартных структур данных, и весьма замысловато устроенная. Она сложная, она эффективная, она обладает встроенным race condition детектором… Неужели не любопытно?!  
Самый популярный неправильный ответ: «Это хеш-таблица». Да, это хеш-таблица. Как устроена хеш-таблица?  
Наводящие вопросы: какая hash-функция используется в map в Go? Что такое bucket?   
  
7. Каков порядок перебора map?  
  
  
Что я хочу оценить: понимает ли кандидат, как отражается устройство структуры данных на ее свойствах. Ну или — читал ли кандидат документацию на базовые типы и запомнил ли неочевидно-важное из нее…   
Самый популярный неправильный ответ: «В порядке вставки». Хеш-таблица, которая сортирует элементы в порядке вставки, ага. А как в ней происходит выборка по ключу в таком случае?  
Наводящие вопросы: как получить одно случайное значение из map?  
  
8. Что будет, если читать из закрытого канала?  
  
  
Что я хочу оценить: читал ли кандидат документацию или сразу бросился кодить. Если читал — запомнил ли очевидно-важное.   
Самый популярный неправильный ответ: «Вернется ошибка». Ну да, ну да… Вы помните синтаксис чтения из канала? Интересно, что синтаксис помнят почти все, но вопрос: «И как вернется ошибка?», — зачастую вводит кандидата в ступор.  
Наводящие вопросы: сколько значений возвращает одно чтение из канала? А почему range-чтение из канала возвращает одно?  
  
9. Что будет, если писать в закрытый канал?  
  
  
Что я хочу оценить: читал ли кандидат документацию или сразу бросился кодить. Если читал — запомнил ли очевидно-важное.   
Самый популярный неправильный ответ: «Вернется ошибка». Да, вернется. Как она это сделает?  
Наводящие вопросы: можно ли закрывать канал со стороны читателя? А если очень надо — как быть?  
  
10. Как вы отсортируете массив структур по алфавиту по полю Name?  
  
Что я хочу оценить: насколько кандидат склонен к написанию «велосипедов».  
Самый популярный неправильный ответ: «Методом пузырька». Пузырек — прекрасный алгоритм, но есть сегодня и поэффективнее. Например — quicksort. Не можете с ходу имплементировать quicksort? Так ведь и не надо!  
Наводящие вопросы: какой стандартный пакет предназначен для сортировки любых слайсов? И заодно — как сделать из массива слайс? Отсортируется ли массив при сортировке слайса?  
  
11. Что такое сериализация? Зачем она нужна?  
  
  
Что я хочу оценить: глубину осознания связи простых повседневно используемых приемов с глобальными задачами разработки.  
Самый популярный неправильный ответ: «Для превращения бинарных данных в текстовые». Строго говоря, не всегда этот процесс можно считать сериализацией, и уж точно это не единственное её применение.  
Наводящие вопросы: почему нельзя для сериализации какой-либо переменной просто взять дамп занимаемой ею памяти?  
  
12. Сколько времени в минутах займет у вас написание процедуры обращения односвязного списка?  
  
  
Что я хочу оценить: помнит ли кандидат институтский курс информатики. А если серьезно — это золотой вопрос программистского собеседования. Если бы мне позволили, я бы задавал его первым, и 70% интервью можно было бы на этом месте заканчивать. Один мой друг, который работает в JetBrains, так и поступает, кстати. Так вот, связанный список — это один из базовых элементов computer science, и один раз узнав, как он устроен, забыть это невозможно. Если «программист» не может развернуть односвязанный список, это свидетельствует о серьёзных пробелах в базовых знаниях. И это повод насторожиться — а что еще из базового он пропустил?.  
Самый популярный неправильный ответ: «А что такое односвязанный список?». Это такая структура данных...  
Наводящие вопросы: какие тесты вы бы написали для своей процедуры разворачивания односвязного списка?  
  
13. Где следует поместить описание интерфейса: в пакете с реализацией или в пакете, где этот интерфейс используется? Почему?  
  
  
Что я хочу оценить: степень готовности кандидата использовать средства реализации модульной архитектуры, предлагаемые Go.  
Самый популярный неправильный ответ: «Рядом с реализацией». Вариантов 2, и кандидаты выбирают, похоже, наугад. На вопрос: «Почему?» — ответить затрудняются.  
Наводящие вопросы: что такое tight coupling? Почему это плохо? В каком варианте связанность слабее?  
  
14. Предположим, ваша функция должна возвращать детализированные Recoverable и Fatal ошибки. Как это реализовано в пакете net? Как это надо делать в современном Go?  
  
  
Что я хочу оценить: насколько глубоко кандидат осмыслил эту непростую тему — обработку ошибок в Go.  
Самый популярный неправильный ответ: «Я не знаю». Обработка ошибок в Go одновременно и проста, и сложна. Проста потому, что в крайней степени тупа. Сложна потому, что — до недавнего времени, — любая дифференцированная обработка ошибок не была стандартизована, и каждый справлялся, как мог. Ситуация изменилась, и знать об этом — прямая обязанность ответственного разработчика.  
Наводящие вопросы: что нового и важного в плане обработки ошибок появилось в Go 1.13? А чего не появилось, хоть мы и ждали?  
  
15. Главный недостаток стандартного логгера?  
  
  
Что я хочу оценить: критерии выбора кандидатом 3-party зависимостей для проекта. Логгер — просто самый одиозный случай, когда стандартная библиотека не предоставляет нам необходимой функциональности.   
Самый популярный неправильный ответ: «Я не пользуюсь стандартным логгером». И никто не пользуется, сюрприз! Но почему?  
Наводящие вопросы: каково основное применение информации из логов?  
  
16. Есть ли для Go хороший orm? Ответ обоснуйте.  
  
  
Что я хочу оценить: количество и качество усилий, которые кандидат приложил к выбору инструментов, ускоряющих и облегчающих повседневную деятельность.   
Самый популярный неправильный ответ: «Gorm, вроде, неплох». Конкурирует с ответом «Я всё пишу руками». Я даже согласен с обоими ответами, но давайте обсудим подробнее — как устроен gorm, и почему вдруг руками.  
Наводящие вопросы: что означает буква M в аббревиатуре ORM? Что на эту букву M есть в gorm? А что такое DAL и зачем он нужен?  
  
17. Какой у вас любимый линтер?  
  
  
Что я хочу оценить: уровень знакомства с современными практиками поддержки разработки.   
Самый популярный неправильный ответ: «Встроенный в Goland». Come on!, там нет линтера, там — тривиальный syntax checker!  
Наводящие вопросы: какое отношение линтеры имеют к CI? Зачем нужен CI в процессе разработки?   
  
18. Можно ли использовать один и тот же буфер []byte в нескольких горутинах?  
  
  
Что я хочу оценить: понимает ли кандидат принципы, по которым мы выбираем использовать общую структуру данных или локальную для горутин.  
Самый популярный неправильный ответ: «Можно, если защитить его мьютексом». Я признаю, это плохой вопрос, недостаточно показательный. Но зачем же давать на него хоть и формально правильный, но бесполезный ответ? К сожалению, в рабочей обстановке нам тоже прилетают плохие задачи, но надо уметь адекватно на них реагировать. Адекватно — это не «Что за чушь?!», а «Сформулируйте, пожалуйста, цель», кстати.  
Наводящие вопросы: а зачем и правда может понадобиться использовать один и тот же буфер []byte в нескольких горутинах параллельно? Что именно мы будем в нем защищать мьютексом?  
  
19. Какие типы мьютексов предоставляет stdlib?  
  
  
Что я хочу оценить: насколько глубоко кандидат разобрался в теме конкурентного доступа к данным. Да, по второму разу, но это ключевая тема. На этот раз я захожу со стороны практики.  
Самый популярный неправильный ответ: «Не помню». На самом деле, это ответ: «Не считаю это важным». Ну и зря!  
Наводящие вопросы: что именно и от чего защищает мьютекс?  
  
20. Что такое lock-free структуры данных, и есть ли в Go такие?  
  
Что я хочу оценить: насколько глубоко кандидат разобрался в теме конкурентного доступа к данным.   
Самый популярный неправильный ответ: «Нет».   
Наводящие вопросы: что такое atomic? А что такое sync.Map? Sync.Map — lockfree или нет?  
  
21. Способы поиска проблем производительности на проде?  
  
  
Что я хочу оценить: степень знакомства с этой малоприятной, но постоянно возникающей задачей.   
Самый популярный неправильный ответ: «Пишу в логи». Коллеги, да такие логи сами по себе создают проблемы производительности!  
Наводящие вопросы: какие проблемы производительности вы знаете? Может ли быть так, что потребление ресурсов (CPU, RAM, disk/net bandwidth) вполне умеренное, а пользователи жалуются на «тормоза»? А на что, собственно, жалуются пользователи, и как это связано с тем, что вы видите в системе?  
  
22. Стандартный набор метрик prometheus в Go -программе?  
  
  
Что я хочу оценить: степень готовности использовать лучшие практики современной разработки. И дело тут не столько в самом прометее, сколько в готовности следовать за тенденциями в индустрии. Это, как ни странно, важно — сохранять тесную связь с мейнстримом.  
Самый популярный неправильный ответ: «Я не пользуюсь прометеем». А пора бы, уже несколько лет как пора!  
Наводящие вопросы: оставив прометей в стороне — что вообще Go-программа способна о себе рассказать? Метрики runtime — что это, и откуда берется?  
  
23. Как встроить стандартный профайлер в свое приложение?  
  
  
Что я хочу оценить: на каком уровне кандидату приходилось решать проблемы недостаточной производительности.  
Самый популярный неправильный ответ: «Я не пользуюсь профайлером». Два варианта: или вы никогда не сталкивались с проблемами производительности, или вы по неизвестным причинам игнорируете одно из самых мощных средств борьбы с такими проблемами. И то, и другое снижает вашу ценность как кандидата, и второе сильнее первого.  
Наводящие вопросы: а он нужен, профайлер? Что там есть вообще полезного? И почему его надо встраивать, а не запускать из-под него приложение?  
  
24. Overhead от стандартного профайлера?  
  
  
Что я хочу оценить: тупо — читал ли кандидат документацию. И, если читал, — что понял?  
Самый популярный неправильный ответ: «Ну, заметный».  
Наводящие вопросы: что такое «семплирующий профайлер» и почему это хорошо?  
  
25. Почему встраивание — не наследование?  
  
  
Что я хочу оценить: общее знакомство с ООП-парадигмой и границами ее применения в Go.  
Самый популярный неправильный ответ: «Наследование позволяет переопределять методы». Но, коллеги, встраивание — тоже, тоже!  
Наводящие вопросы: Что означает буква L в аббревиатуре SOLID?  
  
26. Какие средства обобщенного программирования есть в Go?  
  
  
Что я хочу оценить: знакомство с основными парадигмами современного программирования и границами их применимости в Go.   
Самый популярный неправильный ответ: «Интерфейсы». Коллеги, интерфейсы ничего не обобщают (внезапно). Однако программисты, пришедшие в Go с других языков, имеющих развитые средства обобщенного программирования, сильно тоскуют по ним и пытаются эмулировать эти самые средства с помощью интерфейсов. Создают при этом, простите за правду, кошмарных уродцев.  
Наводящие вопросы: что такое map-reduce? Как его реализовать в Go? А без interface{}? А что такое кодогенерация и как ее можно использовать в этой задаче?  
  
27. Какие технологические преимущества языка Go вы можете назвать?  
  
  
Что я хочу оценить: глубину осознания кандидатом места языка Go в современной разработке. Для каких задач Go подходит идеально? Для каких не очень, но будет полезен? Преимущества существуют не сами по себе, а только в контексте задачи!  
Самый популярный неправильный ответ: «Читабельность». Коллеги, читабельность — субъективная категория, она не может быть технологическим преимуществом. Это раз. Два — читабельность Go довольно относительна. Килотонны boilerplate обработки ошибок и ручной раскрутки стека читабельность ни в коем разе не увеличивают!  
Наводящие вопросы: чем отличается goroutine от OS thread? Как устроен сетевой ввод-вывод в Go?  
  
28. Какие технологические недостатки языка Go вы можете назвать?  
  
  
Что я хочу оценить: глубину осознания кандидатом места языка Go в современной разработке. Для каких задач Go совершенно не подходит?  
Самый популярный неправильный ответ: «Отсутствие generic types». Я пробовал спрашивать на этом месте: «Для каких ваших текущих задач были бы полезны генерики?», — и ни разу не услышал ничего внятного. И не услышу — генерики критически важны для функциональных языков, а Go… Об этом ниже.  
Наводящие вопросы: их миллион, но я задам один. Как превратить []io.ReadWriter в []io.Reader?  
  
Всего 28 вопросов, но сколько боли и недоумения они мне принесли! Но и не задавать их я не могу — это моя работа. Эта статья — попытка исправить ситуацию. Возможно, она поможет вам лучше подготовиться следующему собеседованию. Не обязательно по моему опроснику — так или иначе поднятые здесь темы будут затронуты на любом техническом интервью. Спасибо за внимание  
  
Напоследок — обещанное предположение о том, как и почему сложилась эта печальная ситуация. Наша индустрия, как бы мы это ни оценивали, уже давно живет по закону «fail early, fail often, but always fail forward». Для нас это означает, что мы должны делать свою работу как можно… хуже! Мы должны потратить как можно меньше сил, средств и времени, — особенно времени! — на предложенную задачу, ибо никто не знает, имеет ли текущая задача практический смысл. Собственно, большую часть задач мы решаем в режиме разведки, проверяя чужие смелые бизнес-гипотезы.  
  
В результате лучшие кадры приучаются делать только то, что им поручено, ничего, кроме того, что им поручено, и необходимый минимум того, что им поручено. И этот подход, я предполагаю, незаметно и неосознанно эти кадры распространяют и на свое обучение. Разработчик изучает только то, что ему нужно по текущей задаче: ничего, кроме этого, и ограничивается минимально возможным набором знаний.  
  
Моё скромное мнение — так нельзя! Образование — это игра в долгую, и подходить к нему надо с другими критериями, нежели к очередному «Проверим идейку MVP».  
  
 Конференция Golang Live 2020 пройдёт онлайн 14–17 октября. И в этом есть свои плюсы: нам не придётся вводить ограничения на количество гостей, и участником сможет стать каждый желающий. Забронировать билеты можно на все дни конференции, а благодаря генеральному партнеру — компании Юла, — первый день открыт для всех желающих. Смотрите расписание здесь.   
  
Программа конференции раскроет главную тему — «Продуктовая разработка на Gо». Так вы сможете увидеть цельную картину и понять, как Gо работает на реальных (часто высоконагруженных) проектах. А тестирование в случае Go становится ещё интереснее. Сложную бизнес-логику, как правило, нам приходится реализовывать нетривиальным способом, поэтому и писать интеграционные тесты для неё сложно. Что с этим можно сделать и нужно ли все обкладывать интерфейсами и моками, мы и поговорим при встрече.   
  
Пообщаться, задать вопросы (и ответить на вопросы других) вы можете в этом telegram-канале. Посмотреть новости о конференции — в другом, или на выбор: Facebook, VKontakte, Twitter, Youtube, — если вы предпочитаете именно их.  
  
До встречи на Golang Live 2020!