

Codemia / ITØТЕЦ

БАНК ВОПРОСОВ BACKEND (DEMO)

(без привязки к языку и фреймворку)

Эти вопросы задавали мне на собеседованиях, эти вопросы задаю я, когда набираю людей в команду.

Пользуйся на здоровье! Буду рад если отблаговаришь активностью в моем профиле (лайки, комментарии).

ОБЕСЕДОВАНИЯМ НА IT ПОЗИЦИИ ОТ IT ОТЦА
РАНИТЬ В СУХОМ НЕДОСТУПНОМ ОТ HR-ОВ МЕС

Основы производительности:

Зачем бэкенд-разработчику нужно знать о вычислительной сложности?

Чем отличается поток от процесса?

Какие ключевые факторы влияют на производительность программного кода?

Как кэширование данных может ускорить работу программы?

Как использование кэша влияет на скорость?

Почему выделение памяти в куче медленнее, чем в стеке?

В каких случаях inline-функции ускоряют выполнение программы?

Чем стек отличается от кучи с точки зрения скорости работы?

```
console.log("Yay! " + successMessage)
});const myFirstPromise = new Promise((resolve, reject) => {
// We call resolve(...) when what we were doing asynchronously was successful, and
// set a timeout for when we should reject.
resolve("Everything went well!");
}, 250)
})

myFirstPromise.then((successMessage) => {
// successMessage is whatever we passed in the resolve(...) function above.
// It doesn't have to be a string, but if it is only a success message, it probably
console.log("Yay! " + successMessage)
```

Параллельность и многопоточность:

Чем потоки отличаются от процессов в контексте производительности?

Какой overhead возникает при переключении контекста между потоками?

Какие преимущества дает hyper-threading?

Какие механизмы синхронизации замедляют работу многопоточных программ?

Чем корутины выгоднее потоков при работе с I/O?

Почему блокировка потоков (locking) может вызывать проблемы с производительностью?

Как зелёные потоки (green threads) влияют на производительность?

```

console.log("Yay! " + successMessage)
});const myFirstPromise = new Promise((resolve, reject) => {
  // We call resolve(...) when what we were doing asynchronously was successful, and
  // reject(...) when it failed.
  setTimeout(() => {
    // Everything went well!
    resolve("Success!");
  }, 250)
});

myFirstPromise.then((successMessage) => {
  // successMessage is whatever we passed in the resolve(...) function above.
  // It doesn't have to be a string, but if it is only a success message, it probably
  console.log("Yay! " + successMessage)
});
    
```

Производительность в базах данных и системах хранения:

Как индексы в базах данных ускоряют запросы?

Почему нормализация может как улучшать, так и ухудшать производительность БД?

Какие техники кэширования используются для ускорения работы БД?

Какой есть недостаток у ORM?

Чем GUID (UUID) лучше чем классический автоинкрементальный id?

Для чего нужен CQRS? Какие есть плюсы и минусы?

Как уровень изоляции транзакций влияет на производительность БД?

Какие инструменты используются для профилирования запросов в PostgreSQL?

SQL

Какие бывают типы JOIN в SQL?

Чем отличается INNER JOIN от LEFT/RIGHT JOIN?

Что такое CROSS JOIN?

Как работает GROUP BY?

В чем разница между HAVING и WHERE?

Как использовать CASE в SQL-запросах?

Какие бывают подзапросы в SQL?

```
// It doesn't have to be a string, but if it is only a success message, it probably
console.log("Yay! " + successMessage)
});const myFirstPromise = new Promise((resolve, reject) => {
  // We call resolve(...) when what we were doing asynchronously was successful, and
  setTimeout( function() {
    | resolve("Success!") // Yay! Everything went well!
  }, 250)
})

myFirstPromise.then((successMessage) => {
  // successMessage is whatever we passed in the resolve(...) function above.
  // It doesn't have to be a string, but if it is only a succeed message, it probably
  console.log("Yay! " + successMessage)
```

Продолжение в Прайде

[Подпишись тут](#)