Разработчик, пишущий качественное программное обеспечение, но еще не умеющий анализировать, декомпозировать задачи. Пока не умеет видеть стратегическую ценность исполняемой работы. Является качественным исполнителем и поддержкой для Middle разработчика

## Общие вопросы

1. Назовите основные принципы ООП
2. Что такое наследование, инкапсуляция, полиморфизм: привести примеры из собственного опыта. От какого класса неявно наследуются все классы в .NET. Разрешено ли множественное наследование.
3. Что такое рекурсия
4. Что такое лямбда-выражение
5. Что такое параллельное программирование и его назначение. Какие классы используются
6. Что такое JSON
7. Что такое REST
8. С какими GoF-паттернами знакомы (в идеале, если использовали на практике)
9. Какая разница между GET, POST, PUT, DELETE, PATCH (HTTP)методами
10. Знаком ли кандидат с подходом agile/scrum
11. SOLID. Либо рассказать побуквенно, либо может вспомнить библиотеку в дотнет которая идеально иллюстрирует все 5 принципов
12. Вопрос о базовых принципах безопасности приложений со стороны разработчика

Сертификаты, обработка исключений, хэширование паролей, обработка инпутов, пользовательская аутентификация

1. Паттерны: какие паттерны знает кандидат (цель: понять насколько хорошо кандидат панимает причины использования тех или иных паттернов)

## Обработка исключений

1. С помощью какого класса .NET производится обработка исключений
2. Что такое обработка исключений в C#? Как она работает?
3. Как создать свое исключение
4. В чём разница между **throw** и **throw ex** в обработке исключений в C#?

Объясните, что *throw ex* повторно вызывает исключение, сбрасывая трассировку стека, а *throw* сохраняет оригинальную трассировку стека. Сделайте упор на важности сохранения трассировки стека для отладки

1. Чем отличаются следующие виды обработки исключений в блоке catch
2. try
3. {
4. // some exception code
5. }
6. catch(Exception ex)
7. {
8. throw;
9. throw ex;
10. throw new Exception("some message", ex);

}

## Платформа .NET

1. Что такое .NET
2. Что такое CLR
3. Что такое сборщик мусора (достаточно общего описания)
4. Что такое delegate
5. Что такое LINQ и для чего используется
6. Что такое ASP.NET

## Типы данных, коллекции и структуры

1. Чем отличаются значимые и ссылочные типы. (Value vs Reference)
2. String это значимый или ссылочный тип. Если ссылочный, то почему, когда передаем переменную типа String в метод, то оригинал переменной не изменяется
3. В чем отличие между String и StringBuilder
4. Что такое обобщенные типы (generic). Какие проблемы они решают
5. Что такое boxing и unboxing
6. Что такое Array, List, HashSet, Dictionary. Как работают, чем отличаются, когда лучше использовать
7. Если нужно часто добавлять и удалять элементы, то какую коллекцию лучше использовать

## Классы, структуры и интерфейсы

1. Чем отличается класс от абстрактного класса
2. Чем отличаются абстрактный класс от интерфейса.
3. Для чего нужны интерфейсы и какие задачи они выполняют
4. Какие вы знаете модификаторы доступа
5. В чем разница между обычным классом и статическим
6. В чем разница переопределения метода между ключевыми словами new и override
7. Может ли объект структуры храниться в куче

## Асинхронность

1. Что такое асинхронность и чем отличается от многопоточности
2. Что означают ключевые слова async/await

## Базы данных

1. Какая разница между реляционными и не реляционными базами данных.  
   Опциональный вопрос: Плюсы и минусы использования
2. Что такое индексы в реляционных базах данных. Какие индексы существуют
3. Как сгруппировать результат запроса
4. Каким образом можно сделать запрос и получить результат из нескольких таблиц в виде одной таблицы
5. Какие типы Join существуют в SQL и как работают

## Тестирование

1. Для чего нужны юнит-тесты
2. В чем разница между юнит и интеграционными тестами
3. Из каких трех логических блоков состоит юнит-тест (опциональный вопрос. По идее джуниор должен это знать, но вряд ли будет)

## Практика

1. Class A  
   {  
   virtual void Foo()  
   {  
   Console.WriteLine("Class A");  
   }  
   }

Class B: A  
{  
override void Foo()  
{  
Console.WriteLine("Class B");  
}  
}  
Что выведут на консоль следующие вызовы  
B obj1 = new A();  
obj1.Foo();  
B obj2 = new B();  
obj2.Foo();  
A obj3 = new B();  
obj3.Foo();

Middle

Профессиональный разработчик программного обеспечения, умеющий анализировать бизнес требования, декомпозировать задачи, работать совместно с другими членами команды и коммуницировать с представителями бизнеса. Способен видеть стратегическую ценность разрабатываемого продукта и вносить предложения по улучшению. Является основной движущей единицей проекта

## Общие вопросы

1. Предположим у нас есть задача разработать веб апи (просто круды). На реализацию у нас есть около 2-3 недель. Какие бы библиотеки, фреймворки вы бы использовали для разработки. Какой шаблон проекта бы использовали для разработки. После ответа при необходимости можно уточнить, допустим время на разработку было увеличено до месяца. Далее начало обсуждения. И все таки для чего принято разделять слои, почему в комьюнити так сильно порицается если проект не разбит по слоям. Выводим кандидата на разговор об опыте.
2. Как можно организовать глобальные обработчики исключений
3. Вопрос о базовых принципах безопасности приложений со стороны разработчика

Сертификаты, обработка исключений, хэширование паролей, обработка инпутов, пользовательская аутентификация

1. Узнать знаком ли кандидат с полным циклом разработки

Планирование, сбор требований и анализ, дизайн, разработка и внедрение, тестирование, публикация, поддержка

1. **Как вы понимаете принципы SOLID**

S - single responsibility

O - open close

L - Liskov substitude

I - Interface segregation

D - Dependency inversion

1. **Приведите в пример библиотеку в .net подходящую под определение SOLID** (Entity Framework)
2. **Что такое инъекция зависимостей и зачем оно нужно**

Стиль настройки объекта, при котором поля объекта задаются внешней сущностью. Другими словами, объекты настраиваются внешними объектами

1. **Чем инъекция зависимостей отличается от инверсии зависимостей**

Разделение зависимости высшего уровня от низшего

1. **С какими паттернами проектирования вы знакомы?**
2. **Для чего нужен паттерн Strategy**

Шаблон проектирования, который определяет набор алгоритмов, инкапсулирует каждый из них и обеспечивает их взаимозаменяемость

1. **Какие ключевые различия между распределенными системами и монолитными**
2. **Какие паттерны проектирования распределенных систем вы знаете**

CQRS

2PC

SAGA

Gateway

Sharding

1. **Расскажите, как работают асинхронные методы**

**Асинхронность** позволяет вынести отдельные задачи из основного потока в специальные асинхронные методы и при этом более экономно использовать потоки. Асинхронные методы выполняются в отдельных потоках. Однако при выполнении продолжительной операции поток асинхронного метода возвратится в пул потоков и будет использоваться для других задач. А когда продолжительная операция завершит свое выполнение, для асинхронного метода опять выделяется поток из пула потоков, и асинхронный метод продолжает свою работу.

## Платформа .NET

1. **Объясните разницу между отложенным и немедленным исполнением в LINQ.**

При отложенном выполнении LINQ-выражение не выполняется, пока не будет произведена итерация или перебор по выборке, например, в цикле foreach. Обычно подобные операции возвращают объект IEnumerable<T> или IOrderedEnumerable<T>.

С помощью ряда методов мы можем применить немедленное выполнение запроса. Это методы, которые возвращают одно атомарное значение или один элемент или данные типов Array, List и Dictionary

1. **Какая разница между services.AddTransient и services.AddScope в ASP.NET Core**

AddTransientTransient подразумевает, что сервис создается каждый раз, когда его запрашивают. Этот жизненный цикл лучше всего подходит для легковесных, не фиксирующих состояние, сервисов.

AddScopedScoped - сервис создаются единожды для каждого запроса.

AddSingletonSingleton - сервис создается при первом запросе (или при запуске ConfigureServices, если вы указываете инстанс там), а затем каждый последующий запрос будет использовать этот же инстанс.

## Управление памятью

1. **Что такое куча и стек**

Стек — это область оперативной памяти, которая создаётся для каждого потока. Он работает в порядке LIFO (Last In, First Out),  то есть последний добавленный в стек кусок памяти будет первым в очереди на вывод из стека. Каждый раз, когда функция объявляет новую переменную, она добавляется в стек, а когда эта переменная пропадает из области видимости (например, когда функция заканчивается), она автоматически удаляется из стека. Когда стековая переменная освобождается, эта область памяти становится доступной для других стековых переменных.

Куча — это хранилище памяти, также расположенное в ОЗУ, которое допускает динамическое выделение памяти и не работает по принципу стека: это просто склад для ваших переменных. Когда вы выделяете в куче участок памяти для хранения переменной, к ней можно обратиться не только в потоке, но и во всем приложении. Именно так определяются глобальные переменные. По завершении приложения все выделенные участки памяти освобождаются. Размер кучи задаётся при запуске приложения, но, в отличие от стека, он ограничен лишь физически, и это позволяет создавать динамические переменные.

1. **Зачем нам зарезервированное слово using в C#, если в .NET есть автоматическое управление памятью**

Самым большим ограничением сборщика мусора в .NET является то, что хотя он и преподносится как способный обнаруживать и удалять неиспользуемые объекты, на самом деле он находит только те объекты, на которые отсутствуют ссылки

Есть и ресурсы, которые не попадают в эту категорию, например, дескрипторы файлов должны быть закрыты как можно быстрее, чтобы не вызвать конфликтов между приложениями. Такие ресурсы не могут полностью управляться сборщиком мусора, поэтому .NET предоставляет метод Dispose() вместе с конструкцией using() для объектов, управляющих этими ресурсами. Дефицитные ресурсы, используемые объектом, быстро освобождаются реализацией метода Dispose() вручную (или используя метод using(), что также можно считать ручным освобождением)

## Базы данных

1. **Расскажите о нормальных формах в СУБД**
2. **Что такое индекс в БД**
3. Когда следует использовать индексы? Преимущества и недостатки
4. Какие типы индексов существуют
5. Что такое ACID
6. Какие вы знаете уровни изоляции транзакций
7. Проблема: запрос долго выполняется. Какие есть методы ее диагностики и решения
8. Как подключиться к базе данных в С#?
9. Что такое Entity Framework в C#?

## Практические вопросы

1. Спроектируйте базу данных медицинского центра. Не нужно уходить в реализацию, достаточно описать схематично и дать верхнеуровневое описание схемы.  
   Цель: Ваша задача спроектировать базу данных портала медицинского центра, чтобы клиенты могли выбирать врача к которому хотят записаться на прием, видеть график работы выбранного врача и записаться на приме по системе электронной очереди.
2. Как бы вы проверили коллекцию на наличие дубликатов
3. Имеются три сервиса. Выполняется HTTP-запрос на первый. Первый должен записать данные во второй, а второй - в третий. Как гарантировать, что данные при такой сложной коммуникации не потеряются? Как предотвратить записи дубликатов данных?
4. Что выведет на консоль следующий код  
   var s1 = string.Format("{0}{1}", "abc", "cba");

var s2 = "abc" + "cba";

var s3 = "abccba";

Console.WriteLine(s1 == s2);

Console.WriteLine((object)s1==(object)s2);

Console.WriteLine(s2==s3);

Console.WriteLine((object)s2==(object)s3);

Senior

## Общие вопросы

1. Какие ещё практики, кроме ООП использовали
2. Работали с предметно ориентированным программированием
3. Что такое слабосвязанный код? Чем он лучше сильносвязанного кода? Как бы вы достигали более слабой связности кода?
4. Как работает async / await (подробно)? Почему нельзя использовать async void методы?

## Платформа .NET

1. Как работает lock? Можно ли использовать структуры внутри выражения lock
2. Что такое Expression Tree
3. Как работает сборщик мусора (подробно)? Почему в GC три поколения, а не, скажем, пять, десять или два?
4. Как бы вы организовали трассировки Web API сервисов?
5. Как в .NET Core можно настроить хранение секретов на компьютерах разработчиков и на рабочих средах?

## Типы данных и коллекции

1. Как создать собственный immutable-тип?
2. Какие структуры данных вы реализовывали сами для платформы .NET? Расскажите, чем они отличались от стандартных реализаций.

## Базы данных

1. Как БД сохраняет данные
2. Что такое денормализации
3. Когда и какие уровни изоляции транзакций можно использовать
4. Как в популярных СУБД реализованы принципы ACID
5. Приходилось ли вам оптимизировать запрос в БД? Если да, то как?
6. Опишите, какие вы знаете потенциальные проблемы, связанные с параллельными запросами к БД.
7. Как предотвратить конкурентные записи

## Архитектуры

1. Какую проблему решают микросервисы
2. Какие есть способы коммуникации микросервисов
3. Как бы вы решили проблему межсервисной транзакционности
4. В чем разница между оркестрацией и хореографией
5. Расскажите варианты реализации распределенных транзакций в микросервисах
6. Каким образом вы будете налаживать систему, состоящую из множества микросервисов, если нужно отследить полный путь обработки запроса
7. Чем отличаются синхронные и асинхронные сервисы
8. Предположим у клиента моргнул свет и сервис вернулся со статусом ошибки, хотя операция была произведена успешно. Стоит ли клиенту сделать повторный запрос?

## Тестирование и модульное тестирование в C#

1. Что такое модульное тестирование и как его провести в C#?
2. Что такое разработка через тестирование (TDD)?

## Лучшие практики для C#

1. Каковы лучшие практики для написания чистого и поддерживаемого кода на C#?

Как ответить: Упомяните следующие практики: использование последовательных соглашений об именах, использование комментариев и документации, написание модульного и повторно используемого кода, оптимизация производительности, следование принципам SOLID.

## Практические вопросы

1. Создайте клон любой популярной социальной сети (рисуем дизайн системы и обсуждаем)
2. Допустим, есть сущность сделки (Bid). И в сделке есть список партнеров, с которыми эта сделка заключается. Сами сделки хранятся в сервисе сделок (bidding service), а партнеры - в своем сервисе (partners service). И когда показывается список сделок, на начальном скрине нужно отразить количество партнеров. Как бы вы спроектировали хранение и изображение этого счетчика?