Компетентності рівня junior для .Net Developer	**Загальна оцінка компетентності (max. 6)**	**GlobalLogic**	**AltexSoft**
1. Основи C# та .NET			
1.1 Архітектура .NET Знає основні компоненти архітектури .NET, такі як Common Intermediate Language (CIL), Assembly, Common Language Runtime (CLR), Common Language Infrastructure (CLI), Common Type System (CTS), JIT-компілятор, Framework Class Library (FCL) та Base Class Library (BCL). Розуміє, як ці компоненти взаємодіють між собою для виконання коду, керування пам'яттю, забезпечення безпеки та обробки винятків. Вміє використовувати ці знання для створення ефективних та надійних додатків на платформі .NET.	**4**	2	2
1.2 Змінні та типи даних. Структури даних Знає основні типи даних в C# (int, float, double, bool, string, тощо) та їх використання. Вміє оголошувати і ініціалізувати змінні різних типів. Розуміє роботу з рядками, масивами та колекціями. Розуміє різницю між статичними та динамічними структурами даних. Вміє вибирати відповідну структуру даних для конкретного завдання. Знає особливості роботи з nullable types, вміє обробляти null-значення та використовувати оператор ?? (null-coalescing operator) для запобігання помилкам.	**5**	2	3
1.3 Оператори та цикли, інструкції мови, умовні конструкції Знає основні оператори та цикли в C# (арифметичні, логічні, порівняння, присвоювання) і вміє застосовувати їх для виконання різноманітних обчислень та умовних операцій. Розуміє принципи роботи умовних конструкцій (if, else if, else, switch) і може використовувати їх для реалізації логічних розгалужень у програмному коді. Вміє застосовувати цикли (for, while, do-while, foreach) для ітерації по масивах та колекціях, створення повторюваних дій і обробки даних.	**5**	2	3
1.4 Функції. Знає, як створювати та використовувати функції в C#. Розуміє основні концепції, такі як визначення функцій, передача параметрів і повернення значень. Вміє використовувати перевантаження методів та рекурсію, розуміє важливість та способи використання локальних та глобальних змінних у функціях. Здатний застосовувати функції для структуризації коду, покращення його читабельності та повторного використання. Вміє працювати з анонімними методами та lambda-виразами, а також використовувати делегати для передавання методів як параметрів.	**5**	2	3
1.5 Основи структури коду Розуміє, як працювати з автоматичними властивостями (get та set), що дозволяють швидко створювати властивості класу без додаткового коду. Знає, як організовувати код за допомогою просторів імен (namespaces) для логічного групування класів і уникнення конфліктів імен. Розуміє, як використовувати псевдоніми (aliases) для скорочення та спрощення написання коду, що покращує його читабельність та організацію	**5**	2	3

1.6 Помилки та виключення (Errors & Exceptions) Знає, що таке null значення, та як воно використовується для позначення відсутності об'єкта або значення. Вміє перевіряти змінні на null за допомогою умовних операторів (if (variable == null)), щоб уникнути помилок часу виконання. Знає та використовує оператор ?? для надання значення за замовчуванням, якщо змінна дорівнює null. Вміє використовувати оператор ?. для безпечного доступу до членів об'єкта, який може бути null. Розуміє, як працювати з null-сполученими типами (nullable types) для значимих типів даних (наприклад, int?, double?). Знає, що таке виключення (exception), та чому і коли воно виникає. Знає про найпоширеніші типи виключень, такі як NullReferenceException, InvalidOperationException, ArgumentException, та вміє їх розрізняти. Вміє вручну генерувати виключення за допомогою оператора throw (наприклад, throw new ArgumentNullException("parameterName")). Вміє використовувати блоки try-catch для обробки виключень та запобігання аварійного завершення програми. Вміє додавати блок finally для виконання коду, який має виконатися незалежно від того, виникло виключення чи ні. Розуміє, як створювати власні типи виключень, що успадковуються від базового класу Exception, для специфічних ситуацій у програмі.	**5**	2	3
2. Теоретичний блок. Розуміння ООП			
2.1 Основні концепції ООП. Класи та об'єкти Знає принципи використання об'єктів для організації та структурування даних та функціональності у програмі. Ідентифікує розрізнення між класом (шаблоном) та його екземплярами (об'єктами). Розуміє основні концепції ООП (інкапсуляція, спадкування та поліморфізм) Вміє ініціалізувати об'єкти та працювати з конструкторами для встановлення початкового стану об'єктів.	**5**	2	3
2.2 Класи, конструктори, властивості, модифікатори доступу Знає, що таке клас, його роль в об'єктно-орієнтованому програмуванні (ООП). Вміє створювати класи, описувати їх властивості та методи. Розуміє, для чого потрібні конструктори, як вони працюють. Знає про конструктори за замовчуванням, параметризовані конструктори. Вміє створювати та ініціалізувати об'єкти з використанням конструкторів. Вміє створювати автоматичні властивості з get та set методами. Знає про основні модифікатори доступу (private, protected, internal, public) та вміє їх застосовувати для контролю доступу до членів класу. Розуміє, що таке анонімні типи, як і коли їх використовувати. Вміє створювати анонімні типи для зберігання тимчасових даних.Знає, що таке extensions methods, як вони працюють. Вміє створювати mextensions methods для додавання нової функціональності до існуючих класів без їх зміни.	**5**	2	3
2.3 Інкапсуляція Знає основи інкапсуляції як одного з основних принципів об'єктно-орієнтованого програмування. Це включає розуміння того, як приховувати внутрішню реалізацію класів, надаючи доступ до даних через публічні методи та властивості. Вміє використовувати модифікатори доступу (private, public, protected) для контролю видимості членів класу, що дозволяє запобігти випадковій зміні стану об'єкта ззовні. Має розуміння, як інкапсуляція сприяє підтримці коду, його безпеці та зручності в обслуговуванні.	**5**	2	3
2.4 Наслідування Знає основи наслідування в об'єктно-орієнтованому програмуванні, розуміє, як використовувати базові та похідні класи для створення ієрархій класів і повторного використання коду. Має вміти перевантажувати оператори та методи, щоб налаштувати їхню поведінку під час роботи з об'єктами своїх класів. Знає, як застосувати часткові методи (partial methods), для чого вони призначені та як їх використовувати для розширення функціональності класів без зміни їхньої структури.	**5**	2	3
3. Розширені знання .NET			
3.1 Робота з delegate, event та lambda expressions. Розуміє базові концепції делегатів, подій та lambda виразів. Вміти використовувати делегати типу Func, Action та Predicate для створення та виклику методів, що передаються як параметри. Розуміє, як працюють події та як їх використовувати для обробки асинхронних або користувацьких дій. Знає, як створювати та застосовувати анонімні методи і lambda вирази для спрощення коду та підвищення його читабельності. Вміє використовувати ці інструменти для вирішення простих задач у повсякденній розробці.	**4**	2	2

3.2 Узагальнені та не узагальнені колекції Розуміє особливості роботи з колекціями в .NET, включаючи застосування просторів імен System.Collections та System.Collections.Generic. Знайомий з основними класами, такими як ArrayList, SortedList, Hashtable, Queue, Stack, а також узагальненими колекціями, такими як List<T>, Queue<T>, Stack<T> і Dictionary<T, V>. Роуміє принципи роботи з інтерфейсами IList, ICollection, IEnumerable, та IDictionary, а також вміє використовувати ітератори та оператор yield для створення ітеративних методів. Має базові знання про узагальнені типи (generic types) та їх переваги, такі як типова безпека та покращена продуктивність. Вміє обирати правильний тип колекцій залежно від задачі та оптимізувати їх використання.	**5**	2	3
3.3 Використання кортежів Знає основи роботи з кортежами (клас Tuple) у C#. Вміє створювати та використовувати кортежі для зберігання кількох значень різних типів в одному об’єкті без необхідності створювати окремий клас. Розуміє, як ініціалізувати кортежі, отримувати доступ до їхніх елементів, і використовувати їх для передачі декількох значень між методами.Володіє навичками створення та розпакування кортежів, використання іменованих кортежів для більшої зручності та читабельності коду. Має розуміння, в яких випадках використання кортежів є доцільним для спрощення коду та підвищення його ефективності.	**3**	2	1
3.4 Робота з потоками та файловою системою Знає основи роботи з файловою системою через простір імен System.IO. Знає, як застосовуються основні класи, такі як Directory, File, FileInfo та DirectoryInfo, для виконання операцій з файлами та папками, включаючи створення, видалення, копіювання та переміщення. Вміє читати та записувати дані за допомогою класів StreamReader та StreamWriter, а також використовувати BinaryReader та BinaryWriter для роботи з бінарними файлами. Знає, як застосовувати класи Path і DriveInfo для роботи з шляхами та інформацією про диски, а також вміє організовувати асинхронну обробку даних з використанням потоків.	**5**	2	3
3.5 Серіалізація Знає основи серіалізації в .NET, включаючи роботу з просторами імен System.Runtime.Serialization та System.Xml.Serialization. Вміє використовувати класи BinaryFormatter, SoapFormatter, XmlSerializer, і DataContractJsonSerializer для перетворення об'єктів у різні формати (бінарний, SOAP, XML, JSON) та їх відновлення. Розуміє, як застосовувати атрибути Serializable і NonSerialized для контролю процесу серіалізації. Вміє створювати і використовувати прості приклади серіалізації та десеріалізації об'єктів, а також розуміє основні принципи безпеки і продуктивності, пов'язані з цими процесами.	**5**	2	3
3.6 Dynamic Language Runtime Знає основи Dynamic Language Runtime (DLR) та особливості використання динамічних типів у .NET. Це включає розуміння, що таке dynamic тип і коли його варто застосовувати для вирішення завдань, що вимагають гнучкості у визначенні типів на етапі виконання програми. Вміє використовувати dynamic для спрощення взаємодії з COM об'єктами, бібліотеками з динамічно типізованих мов (наприклад, Python), а також для обробки даних з JSON або XML форматів. Розуміє переваги та обмеження динамічних типів, включаючи можливі проблеми з продуктивністю та відсутністю перевірки типів на етапі компіляції.	**3**	2	1
3.7 Багатопотокове програмування Знає основи багатопотокового програмування, включаючи особливості використання класів Thread та ThreadPool для створення і управління потоками. Розуміє механізми синхронізації потоків, такі як ключове слово lock, та класи AutoResetEvent, Monitor, Mutex, Semaphore для запобігання конфліктам доступу до спільних ресурсів. Знає, як використовувати клас Timer для планування задач. Має базові знання про бібліотеку паралельних задач (Task Parallel Library, TPL), зокрема, розуміє, як застосовуються класи Task і Parallel. Вміє створювати і виконувати паралельні задачі, використовуючи методи AsParallel та WithDegreeOfParallelism для підвищення продуктивності додатків. Розуміє переваги і обмеження паралельної обробки та знає, як застосовувати TPL для оптимізації обробки даних.	**3**	2	1
3.8 Асинхронне програмування Знає основи асинхронного програмування, включаючи використання асинхронних делегатів та Task-based Asynchronous Pattern (TAP).Розуміє принципи асинхронності для покращення продуктивності додатків, особливо при виконанні операцій вводу-виводу або тривалих обчислень. Вміє використовувати ключові слова async та await для написання асинхронного коду, що дозволяє виконувати задачі без блокування головного потоку. Знає, як працюють асинхронні методи та як їх правильно викликати, обробляти виключення та забезпечувати коректне завершення задач.	**5**	2	3

3.9 Reflection Має розуміння того, що таке рефлексія та володіє основами роботи з рефлексією в .NET, використовуючи простір імен System.Reflection. Вміє працювати з основними класами, такими як Assembly, MemberInfo, PropertyInfo, TypeInfo, MethodInfo, Type, та Activator. Вміє завантажувати та досліджувати збірки, отримувати інформацію про типи та їх члени (властивості, методи, поля), а також створювати екземпляри типів динамічно під час виконання програми. Знає, як використати рефлексію для доступу до метаданих і динамічної взаємодії з об'єктами.	**3**	2	1
3.10 Expression trees Має розуміння того, як створювати, компілювати та виконувати дерева виразів для динамічної побудови і виконання коду на рівні виразів. Вміти використовувати класи зі простору імен System.Linq.Expressions для створення простих дерев виразів, що представляють вирази лямбда або інші анонімні методи. Розуміє, як дерева виразів використовуються в LINQ для трансляції запитів в SQL або інші форми коду, а також знає основні сценарії їх використання для динамічного створення і виконання коду. Використовує методи Expression, таких як Parameter, Constant, Lambda, Call та інші для створення та маніпулювання деревами виразів.	**3**	2	1
4. Знання Web-технологій. Основи HTML та CSS, JavaScript			
4.1 Загальні питання веб технологій Розуміє клієнт-серверну архітектуру. Знає основні мережеві протоколи, розуміє принцип роботи протоколу HTTP, його основні методи, статусні коди та їх значення. Вміє працювати з заголовками HTTP, формувати HTTP-запити, вказуючи метод, шлях, заголовки та дані. Може обробляти HTTP-відповіді, читати статусні коди, заголовки та вміст відповіді. Розуміє принципи взаємодії із зовнішніми API з використанням HTTP-запитів Має навички використання інструментів розробника браузера для відстеження запитів мережі та аналізу заголовків та вмісту відповідей. Вміє використовувати cookie, LocalStorage для зберігання даних користувача	**6**	3	3
4.2 Загальна структура HTML-документа. Теги та атрибути. Семантика Знає основні елементи HTML-документу та вміє створювати з них правильну структуру. Знає основні HTML-теги, розуміє їх призначення та особливості, вміє обирати та використовувати найбільш доречні HTML-теги для розмітки різних типів контенту. Використовує семантичні елементи для більш чіткого та структурованого опису вмісту вебсторінки. Розуміє роль та значення атрибутів та їх вплив на поведінку та відображення елементів, вміє додавати та налаштовувати атрибути для елементів HTML. Розуміє важливість валідного HTML-коду та його ролі для сумісності, доступності та оптимізації вебсторінки. Здатний перевіряти та виправляти помилки валідації HTML за допомогою валідаторів та інструментів розробника.	**5**	3	2
4.3 Основні CSS-властивості та селектори. CSS-анімації. Flexbox. Grid Розуміє принципи каскадності та спадкування стилів. Вміє працювати з css-властивостями, css-селекторами, та їх ієрархією. Ідентифікує поняття блокової моделі, потоку, позиціонування. Має досвід використання CSS-технік flexbox та grid, вміє налаштовувати гнучку сітку для різних екранів та пристроїв, використовує медіазапити для створення адаптивних вебсторінок. Працює з різними одиницями вимірювання	**5**	3	2
4.4 Основи JavaScript Знає синтаксис JavaScript, змінні, оператори та умовні конструкції. Ідентифікує основні поняття, такі як функції, масиви та об'єкти. Вміє маніпулювати DOM (Document Object Model) за допомогою JavaScript, змінювати вміст та стилі елементів, додавати та видаляти елементи. Може здійснювати обробку подій та взаємодію з користувачем. Розуміє процес здістнення асинхронних запитів до сервера за допомогою AJAX (Asynchronous JavaScript and XML). Використовує вбудовані функції JavaScript, такі як fetch(), для отримання та відправлення даних на сервер.	**4**	2	2

4.5 Розуміння роботи API (типи, логіка роботи REST-застосунків) Знає основні принципи REST, такі як ресурси, URL-шляхи, HTTP-методи та параметри запиту. Розуміє структуру URL-шляхів та їх використання для ідентифікації ресурсів і підресурсів. Володіє навичками читання та створення даних у форматах JSON і XML, ідентифікує переваги та недоліки кожного формату і вміє обирати відповідний формат для конкретного випадку використання. Здатний використовувати різні методи автентифікації, такі як токени, сесії та OAuth. Вміє налаштовувати та використовувати механізми автентифікації та авторизації у REST-застосунках. Розуміє рівні доступу та особливості використання ролей користувачів для обмеження доступу до ресурсів	**4**	2	2
4.6 Створення вебдодатків на платформі ASP.NET. Архітектура MVC. Знає основи платформи ASP.NET та архітектури MVC (Model-View-Controller). Вміти створювати прості вебдодатки, використовуючи ASP.NET, організовувати код відповідно до концепції MVC, де логіка програми, представлення користувацького інтерфейсу та управління даними розділені на окремі компоненти. Вміє працювати з контролерами для обробки запитів, моделями для управління даними та представленнями для відображення інформації. Знає основи маршрутизації, налаштування та деплойменту простих вебдодатків на ASP.NET.	**5**	2	3
5. Робота з даними			
5.1 Основи роботи з LINQ та Parallel LINQ для операцій з даними. Знає базові концепції та синтаксис LINQ (Language Integrated Query), включаючи роботу з різними джерелами даних (масиви, колекції, бази даних). Розуміє принципи запитів LINQ для вибірки, фільтрації, проекції, сортування та групування даних. Вміє використовувати стандартні оператори LINQ, такі як Select, Where, OrderBy, GroupBy, Join, та Aggregate. Здатний створювати складні запити для вирішення реальних завдань. Має базові знання про Parallel LINQ (PLINQ) для підвищення продуктивності обробки даних. Знає, як використовувати PLINQ для виконання паралельних запитів, розуміє переваги та обмеження паралельної обробки. Вміє застосовувати методи AsParallel, WithDegreeOfParallelism, ForAll для оптимізації обробки великих наборів даних. Розуміє, як запити LINQ транслюються у відповідні команди SQL (при роботі з базами даних) або як обробляються в пам'яті (при роботі з колекціями). Знає, як перевіряти продуктивність запитів та оптимізувати їх.	**4**	2	2
5.2 Робота з базами даних SQL. SQL Server та SQLite Має базові знання SQL syntax, вміє будувати запити (queries) для отримання інформації з бази даних, модифікації даних та виконання інших операцій. Вміє працювати з таблицями та схемами, розуміє зв'язки між таблицями. Здатний працювати з ключами та індексами для забезпечення ефективного доступу до даних. Розуміє типи з'єднань (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN) та може використовувати їх для об'єднання даних з різних таблиць та отримання з'єднаних даних Може будувати та виконувати SELECT-запити для отримання певних даних з таблиць та фільтрацію даних на основі певних умов, здійснювати їх агрегацію та групування Вміє працювати зі схемами баз даних, встановлювати права доступу користувачів тощо Вміє створювати базові структури баз даних, виконувати міграції та налаштовувати базу даних для потреб проєкту. Знає основи роботи з транзакціями та може забезпечити цілісність даних під час виконання SQL-запитів.	**6**	3	3
5.3 Використання Entity Framework. Знає, як створювати моделі даних (Entity Classes) та взаємодіяти з базою даних за допомогою EF, включаючи створення запитів (LINQ to Entities) для отримання, вставки, оновлення та видалення даних. Розуміє основні концепції EF, таких як Code First або Database First підходи, а також уміє налаштовувати конфігурацію підключення до бази даних.	**4**	2	2

5.4 Використання ORM. Підходи Database First, Code First, Model First. Знає основні принципи об'єктно-реляційного відображення (ORM) і його застосування для роботи з базами даних у .NET. Розуміє різницю між підходами Database First, Code First, та Model First і може обирати відповідний підхід залежно від вимог проєкту. Вміє налаштовувати та використовувати ORM для створення, читання, оновлення та видалення (CRUD) даних у базах даних. Має навички створення моделей даних, генерації баз даних з моделей і навпаки, а також налаштування міграцій для синхронізації змін у схемах баз даних.	**4**	2	2
5.5 Патерни Repository, UnitOfWork and Specification Знає основні концепції та принципи патернів Repository, UnitOfWork та Specification. Розуміє, як ці патерни сприяють розділенню бізнес-логіки та доступу до даних, а також полегшують підтримку та тестування коду. Вміє реалізовувати ці патерни в проєктах, створюючи та використовуючи класи Repository для доступу до бази даних, класи UnitOfWork для керування транзакціями, та класи Specification для визначення критеріїв вибору даних.	**4**	2	2
6. Інструментарій розробника			
6.1 Системи контролю версій - GIT Розуміє git flow - методологію роботи з гілками для ефективного управління процесом розробки. Здатний створювати репозиторії та ініціалізувати проєкти з використанням Git. Розуміє та використовує команди Git (commit, push, pull, fetch, checkout, add, status, revert та інші). Вміє працювати з гілками (branches) для розробки функціональності відокремлено від основної гілки (наприклад, розробка нових функцій у власних гілках та їх об'єднання в основну гілку). Знайомий з процедурою створення пулл-реквестів для обговорення та злиття змін з різних гілок	**6**	3	3
6.2 Використання Visual Studio та SDK ASP.NET Core. Знає основні можливості та функціонал Visual Studio, включаючи створення, налагодження та розгортання проектів. Розуміє, як використовувати інтегровані інструменти для підвищення продуктивності та якості коду, такі як IntelliSense, Code Analysis, Debugger, та інші. Знає, як встановлювати та налаштовувати SDK ASP.NET Core для розробки веб-додатків. Вміє створювати веб-додатки на базі ASP.NET Core, розуміє основи архітектури MVC, вміє налаштовувати маршрутизацію, створювати контролери, моделі та представлення. Здатний використовувати вбудовані можливості ASP.NET Core для обробки запитів, автентифікації, авторизації та роботи з даними.	**5**	3	2
7. Розмовна англійська			
7.1. Правильна побудова фрази з узгодженням часу Вміє використовувати відповідні часові форми дієслів і структури речень для точного вираження часу в комунікації. Здатний правильно використовувати часові форми, такі як Present Simple, Present Continuous, Past Simple, Past Continuous, Future Simple, для передачі правильного значення в повідомленні.	**5**	3	2
7.2 Професійна it термінологія в розмові з замовником Знає і використовує професійні IT-терміни та скорочення, які використовуються в IT-індустрії, для чіткого та зрозумілого спілкування зі замовником. Вміє перекладати складні технічні концепції на доступну мову для замовника, не знайомого з IT-галуззю.	**6**	3	3
7.3 Професійна it термінологія у діловому звіті Знає і використовує спеціалізовану IT-термінологію та фразеологію при написанні ділових звітів. Вміє передати інформацію про технічні питання та проєкти зрозуміло та конкретно, використовуючи адекватні IT-терміни та абревіатури.	**6**	3	3
7.4 Професійна термінологія для .Net Development Знає та розуміє специфічну термінологію, що використовується в контексті .Net Development Розуміє та використовує специфічні терміни, патерни, алгоритми, фреймворки, бібліотеки та інші інструменти, які використовуються в .Net Development	**6**	3	3

7.5 Неформальне спілкування (small talks) Вміє вести неформальну розмову, розпочинати діалоги та підтримувати невимушену атмосферу. Знає загальновживану термінологію та фрази, які використовуються у неформальних розмовах, таких як загальні питання про погоду, цікаві події, особисті захоплення, спорт, фільми тощо.	**5**	3	2

Коментарі до таблиці

Оцінки компетентностей вказані станом на 2024/2025 рр., по шкалі від 0 до 3.

«3» - компетентність вкрай важлива, якщо кандидат нею не володіє, його подальше просування на вакансію неможливе.

«2» - компетентність, знання якої обов'язково знадобиться кандидату при подальшому професійному зростанні.

«1» - некритична компетентність, якою можна знехтувати.

«0» - неактуальна компетентність.