Competency Matrix

Parent Page: [Competence Matrices](https://wiki.effective-soft.com/display/ET/Competence%2BMatrices) , see also [Interviews materials](https://wiki.effective-soft.com/display/ET/Interviews%2Bmaterials)

|  |  |  |
| --- | --- | --- |
| **JUNIOR** | **MIDDLE** | **SENIOR** |
| **Common**   1. OOP 2. VCS | **Common**  1. .NET Framework vs .NET Core | **Common**  1. VCS Flows, merge vs rebase, mono repo vs N repos |
| **Principles, patterns, architectures**   1. SOLID, KISS, DRY, YAGNI 2. Patterns:    1. Creational    2. Behavioral    3. Structural 3. DI/IoC (lifetime, factory) | **Principles, patterns, architectures**   1. Patterns:    1. Concurrency 2. REST, RESTful vs SOAP 3. Understanding of current project(s) architectural(s) approaches 4. Understanding of how to start project from a scratch 5. TDD | **Principles, patterns, architectures**   1. Architectural patterns (monolith, N-layer, microservices, SOA, etc.) 2. System design (tool, libraries, frameworks, databases, etc.) 3. Patterns and approaches:    1. Strangler    2. Anti-Corruption Layer    3. Database Per Service VS Shared Database    4. API Composition    5. Event Sourcing    6. Saga    7. API Gateway    8. Blue-Green Deployment    9. Circuit Breaker    10. Log Aggregation    11. Distributed Tracing    12. Ambassador    13. Sidecar 4. Messaging patterns: REST, gRPC, SOAP, AMQP, GraphQL 5. BDD, DDD |
| **Algorithms and data structures**   1. Data structures: array, stack, list/linked list, dictionary/hashset 2. Algorithms: sorting | **Algorithms and data structures**   1. O-notation (performance/memory) 2. Data structures: b-tree, queue | **Algorithms and data structures**   1. Algorithms optimization 2. Data structures: graph 3. Data structures optimal decisions |
| **.NET and C#**   1. Value/Reference types 2. String (intern, interpolation, immutability), StringBuilder 3. Boxing/unboxing 4. LINQ 5. Generic 6. Polymorphism: key words virtual, override, new 7. Extension methods 8. GC (generations), Finalizers (queue)? | **.NET and C#**   1. Closures, lambdas, delegate vs expression, expression tree 2. Collections (HashSet vs HashTable, etc.) 3. Stack vs Heap, LOH/SOH 4. Managed vs unmanaged resources, memory leaks, IDisposable   /IAsyncDisposable   1. Async/await, exceptions handling, ConfigureAwait 2. Thread pool, tasks vs threads vs processes, thread synchronization 3. Dynamic and anonymous types | **.NET and C#**   1. Span related types 2. Async patterns: TAP, EAP, APM 3. Async/await, state machine, synchronization context, CancellationToken, TaskCompetitionS ource 4. PLINQ, TPL, concurrency collections 5. Service Provider, Service Scope Factory, Service Allocator Pattern |
| **ASP.NET**   1. Web API vs MVC 2. ASP.NET (Core || MVC || WebAPI) request lifecycle 3. Routing configuration, approaches 4. Server-side validation | **ASP.NET**   1. Middleware vs Filters 2. Error handling (+global) 3. Identity, Claims 4. Token-based auth 5. Antiforgery, XSS 6. Open API (Swagger) 7. Model Binding | **ASP.NET**   1. Custom Validators 2. Execution Context |

|  |  |  |
| --- | --- | --- |
| **Common Web**   1. HTTP methods. GET vs POST (security/HTTPS) 2. HTTP requests structure: headers, body, etc. 3. Cookies. 4. HTML: document structure, basic tags. 5. CSS: positioning, selectors, media-query. | **Common Web**   1. HTTP(S) methods. PUT vs POST (idempotent) 2. RESTful API (manifest/principles) 3. HTTP multipart 4. Caching: server side, client side, http headers 5. DNS 6. JWT, Refresh tokens 7. Web sockets, long-poling, ajax 8. CORS | **Common Web**   1. HTTP/2.0, HTTP/3 (QUIC, UDP) 2. Web security (cross-site scripting, etc.) |
| **Databases**   1. MS SQL 2. CRUD 3. Normalization forms 4. Primary keys, Foreign keys 5. Indexes 6. JOINs | **Databases**   1. SQL / NoSQL, Json data type in SQL 2. Nested queries, temporary tables 3. Stored procedures, functions 4. Transactions, ACID 5. JOIN types: left/right/inner/full outer 6. Group By, Having, Cursor 7. Indexes: clustered vs unclustered 8. Constraints 9. Query optimization: profiling, execution plan, denormalization | **Databases**   1. Views 2. Transactions: isolation levels 3. Composite Indexes 4. OLAP & OLTP 5. Scaling: vertical vs horizontal 6. Sharding, partition keys 7. Replication |
| **ORM**   1. IEnumerable vs IQueryable 2. EntityFramework    1. Take/Skip/Select/Group   /Where/Include/OrderBy  /ThenBy   * 1. Navigation properties   2. Fluent API   3. Migrations | **ORM**   1. ADO.NET vs EF vs Dapper vs NHibernate 2. Code first vs Database first 3. EF vs SQL migrations 4. Lazy loading vs Eager loading vs Explicit loading 5. Projection 6. UOF, Repository Patterns | **ORM**   1. Security: sql injections, etc. 2. Bulk operations in EF 3. Optimizations in EF 4. Views and stored procedures in ORM |
| **Testing**   1. Unit testing: AAA 2. Unit vs integration tests 3. Mocking | **Testing**   1. Functional tests 2. Frameworks: xUnit, FluentAssertions, Moq, etc. | **Testing**   1. E2E 2. Cucumber, Specflow 3. Postman, Insomnia, etc. |
| **JavaScript**   1. let, var, const 2. Variable scopes 3. this (call, bind, apply) 4. Hoisting 5. JS Prototypes vs OOP 6. Closure | **JavaScript**   1. Promise vs Observable 2. Virtual DOM 3. Event-Loop 4. Arrow functions 5. Angular/React/Vue 6. JS Modules 7. Babel, polyfils, shim | **JavaScript**   1. JS Generators 2. Memory leak 3. React/Angular/Vue 4. Typescript 5. Web workers, sockets |
|  | **Cloud & DevOps**   1. Azure/AWS/GCP/Hybrid Cloud theoretical knowledges 2. Hosting cloud services (any cloud provider) 3. Storage cloud services (any cloud provider) 4. Database cloud services (any cloud provider) 5. CI/CD pipelines | **Cloud & DevOps**   1. SQL / NoSQL cloud services 2. AMQP services 3. Distributed Cache Services 4. Monitoring & Alerting tools 5. Key Vault 6. Container Registry 7. Release strategy, artifacts, nuget 8. Load balancing, high availability |

|  |  |  |
| --- | --- | --- |
|  | **Containerization & Orchestration**   1. Benefits (Containers vs VM) 2. docker, docker-compose    1. Container (run, build, publish)    2. Volume (named, anonym, host)    3. Network (address, host, ports) | **Containerization & Orchestration**   1. Kubernetes (k8s, k3s, k0s, etc.)    1. Pod, Deployment, Service, ReplicaSet    2. Persistent Volume, Persistent Volume Claims    3. Secret, ConfigMap    4. Helm 2. Windows Containers |
|  | **Methodologies**  1. Scrum working process, meetings, roles, artefacts, limitations | **Methodologies**   1. Scrum vs Kanban/Scrumban vs Waterfall 2. Estimations, Risks 3. Story Point vs Hours; Sprint Velocity 4. Resizing the Scope, Change requests handling, Issues |
| **English:** B1+ | **English:** B1+ | **English:** B2+ |