Interview Q&A for MS Dynamics NAV / Business Central ERP Developer Role

ERP Experience & Business Central

1. **Q: What is your experience with MS Dynamics NAV or Business Central?**
   - **A:** While I don't have direct production experience with MS Dynamics NAV or Business Central yet, I have 8+ months of hands-on ERP development experience building modules for HR, Payroll, Inventory, Invoicing, and Order Processing using C#, .NET Framework, and SQL Server. I'm actively learning C/AL and AL programming languages and studying NAV/Business Central architecture. My strong foundation in C#, SQL, and ERP concepts positions me well to quickly become productive in the Dynamics ecosystem.

2. **Q: Why do you want to transition to MS Dynamics NAV/Business Central development?**
   - **A:** I'm excited about specializing in a leading ERP platform like Dynamics NAV/Business Central. My experience building custom ERP modules has given me deep appreciation for enterprise business processes. I see Dynamics as the natural next step to leverage my C#, .NET, and SQL expertise in a mature, widely-adopted ERP ecosystem. I'm eager to learn C/AL and AL to extend and customize NAV/Business Central for client-specific business needs.

3. **Q: What ERP modules have you worked on?**
   - **A:** I've developed and maintained several core ERP modules including HR Management (employee records, attendance, leave management), Payroll (salary calculation, tax computation, payslip generation), Inventory Management (stock tracking, warehouse management, reorder points), Invoicing (invoice generation, payment tracking), and Order Processing (sales orders, purchase orders, order fulfillment). This experience gives me a solid understanding of business workflows that translate directly to Dynamics implementations.

C/AL and AL Programming

4. **Q: What do you know about C/AL and AL programming languages?**
   - **A:** I understand that C/AL (Client/Server Application Language) is the legacy language for NAV, while AL is the modern language for Business Central with cloud-first design. AL is compiled into extensions, supports Visual Studio Code, and enables source control with Git. Both languages are object-oriented and event-driven. My C# background provides a strong foundation since AL syntax is C-like. I'm currently studying AL development, understanding triggers, events, and the extension-based architecture.

5. **Q: How would you approach learning C/AL and AL coming from a C# background?**
   - **A:** C# and AL share many similarities - both are object-oriented, strongly-typed, and use similar syntax for control structures. I would focus on understanding NAV/BC-specific concepts: tables, pages, codeunits, reports, queries, triggers, and events. I'd start with Microsoft Learn documentation, practice creating simple extensions, study existing customizations, and leverage

my SQL knowledge for table design. My experience with design patterns and OOP principles will transfer directly. I learn quickly and am committed to becoming proficient.

C# & .NET Framework Expertise

6. **Q: Describe your proficiency with C# and the .NET Framework.**
   - **A:** I have 7+ years of professional C# and .NET Framework experience. I'm highly proficient in C# (versions 4.0 through 10), .NET Framework 4.x, and .NET Core. I've built enterprise applications using ASP.NET MVC, Web API, Windows Forms, and WPF. I'm expert in object-oriented programming, implementing SOLID principles, design patterns, and writing clean, maintainable code. I understand the CLR, memory management, and have deep knowledge of the Base Class Library.

7. **Q: What design patterns do you commonly use in C# development?**
   - **A:** I frequently use Repository pattern for data access abstraction, Factory pattern for object creation, Singleton for shared resources, Strategy pattern for interchangeable algorithms, and Dependency Injection for loose coupling. In ERP development, I've used Template Method for business process workflows and Observer pattern for event handling. I choose patterns based on the specific problem, avoiding over-engineering while ensuring maintainability and testability.

8. **Q: Explain the SOLID principles and how you apply them.**
   - **A: S**ingle Responsibility - each class has one reason to change; **O**pen/Closed - open for extension, closed for modification; **L**iskov Substitution - derived classes must be substitutable for base classes; **I**nterface Segregation - clients shouldn't depend on interfaces they don't use; **D**ependency Inversion - depend on abstractions, not concretions. I apply these by creating focused classes, using interfaces for contracts, implementing dependency injection, and avoiding tight coupling.

9. **Q: How do you handle exceptions and error logging in C# applications?**
   - **A:** I use try-catch blocks strategically at boundaries (API controllers, service layers). I create custom exception types for business rule violations, log exceptions with full context (stack trace, user info, parameters) using frameworks like NLog or Serilog. I implement global exception handlers in web applications, return meaningful error messages to users while logging technical details, and use structured logging for easier troubleshooting.

10. **Q: What's your experience with multi-threading and asynchronous programming in C#?**
    - **A:** I use async/await for I/O-bound operations (database calls, API requests) to improve scalability. I understand the difference between Task and Thread, avoid async void except for event handlers, and properly handle cancellation tokens. For CPU-bound operations, I use Task.Run or Parallel.ForEach. I'm careful about thread safety, using locks when necessary, and understand the SynchronizationContext for UI applications.

SQL Server & Database Design

11. **Q: Describe your SQL Server expertise and experience.**

- **A:** I have 7+ years of expert-level SQL Server experience. I'm highly proficient in T-SQL, database design, stored procedures, functions, triggers, views, and performance optimization. I've designed and optimized databases for ERP systems, healthcare applications, and analytics platforms. I understand indexing strategies, query execution plans, transaction management, and database security. I'm comfortable with SSIS for ETL and SSRS for reporting.

12. **Q: How do you approach database design for ERP systems?**

    - **A:** I start with requirements analysis to understand business entities and relationships. I create normalized schemas (typically 3NF) to eliminate redundancy, define primary/foreign keys, and implement referential integrity. I consider data types carefully for storage optimization, create appropriate indexes based on query patterns, and implement audit trails for transactional data. I design for scalability, considering partitioning strategies for large tables, and ensure data consistency through constraints and triggers.

13. **Q: What's the difference between a stored procedure and a function? When do you use each?**

    - **A:** Stored procedures can perform DML operations (INSERT, UPDATE, DELETE), don't return values directly (use OUTPUT parameters or result sets), and can't be used in SELECT statements. Functions must return values, can't perform DML in scalar/inline table-valued functions, and can be used in queries. I use stored procedures for business operations (order processing, data modifications), and functions for calculations and data transformations that need to be embedded in queries.

14. **Q: How do you optimize slow SQL queries?**

    - **A:** I analyze the execution plan to identify bottlenecks (table scans, missing indexes, high cost operations). I add appropriate indexes (check for missing index recommendations), rewrite queries to be set-based rather than row-by-row, avoid functions on indexed columns in WHERE clauses, use EXISTS instead of IN for large datasets, and consider indexed views for complex aggregations. I update statistics, eliminate unnecessary JOINs, and use query hints only when necessary.

Object-Oriented Programming Principles

15. **Q: Explain the four pillars of OOP and give practical examples.**

    - **A: Encapsulation**: Bundling data and methods, hiding internal details (private fields with public properties in C#). **Inheritance**: Creating specialized classes from base classes (Employee base class with Manager, Developer derived classes). **Polymorphism**: Same interface, different implementations (override methods, interface implementations - IPaymentProcessor with CreditCard, PayPal implementations). **Abstraction**: Hiding complexity, exposing only relevant details (abstract classes, interfaces). I use these daily in building maintainable ERP systems.

16. **Q: What's the difference between abstract classes and interfaces in C#?**

    - **A:** Abstract classes can have implementation (concrete methods), constructors, fields, and access modifiers. They support single inheritance. Interfaces define contracts (method signatures), can only have public members (though C# 8+ allows default implementations), and support multiple inheritance. I use interfaces for capability definitions (ILoggable, IExportable) and abstract classes

for shared base functionality with some mandatory overrides (BaseDocument class for Invoice, PurchaseOrder).

17. **Q: Explain inheritance vs composition. When do you use each?**
    - **A:** Inheritance is "is-a" relationship (Manager IS an Employee), while composition is "has-a" (Employee HAS an Address). Inheritance can create tight coupling and fragile base class problems. Composition provides flexibility and easier testing. I prefer composition for most scenarios, using inheritance only for true is-a relationships. For example, in ERP, I'd use composition for Employee having Department reference rather than inheriting from Department.
18. **Q: How do you ensure your code is maintainable and scalable?**
    - **A:** I follow SOLID principles, write small focused methods/classes, use meaningful names, avoid code duplication (DRY), implement proper layering (separation of concerns), write unit tests, use dependency injection for loose coupling, document complex logic, follow consistent coding standards, and conduct code reviews. In ERP systems, I ensure business logic is separate from data access and presentation layers for easier maintenance and testing.

ERP Business Logic & Processes

19. **Q: Describe your understanding of typical ERP business processes.**
    - **A:** ERP systems integrate core business processes: **Finance** (General Ledger, Accounts Payable/Receivable, Fixed Assets), **Supply Chain** (Procurement, Inventory, Warehouse Management), **Sales** (Quotations, Sales Orders, Invoicing), **HR** (Employee Management, Payroll, Attendance), and **Manufacturing** (BOM, Production Planning). These modules share master data and transactional data. I've built modules ensuring data consistency, implementing approval workflows, and maintaining audit trails for compliance.
20. **Q: How do you handle complex business rules in ERP systems?**
    - **A:** I implement business rules in the business logic layer using service classes, keeping them separate from data access and presentation. For validation rules, I use a combination of database constraints, triggers, and application-level validation. For workflow-based rules (approval processes), I implement state machines or workflow engines. I document business rules clearly and make them configurable where possible to accommodate client-specific requirements without code changes.
21. **Q: How would you approach building a new ERP module?**
    - **A:** I start with requirements gathering and business process analysis, create data models (entities and relationships), design normalized database schema, define business logic and validation rules, create data access layer (repositories, stored procedures), implement service layer with business operations, build user interface (forms, grids, reports), implement security and authorization, create test cases for UAT, and document functionality. Throughout, I ensure integration points with other modules are well-defined.

Integration & APIs

22. **Q: Describe your experience building and consuming APIs.**
    - **A:** I've built numerous RESTful Web APIs using ASP.NET Web API and ASP.NET Core. I follow REST principles (resource-based URLs, proper HTTP verbs, status codes), implement authentication/authorization (JWT, OAuth), use proper error handling and validation, version APIs appropriately, and document with Swagger/OpenAPI. I've integrated with third-party APIs (payment gateways, shipping providers, healthcare systems) handling authentication, rate limiting, and error scenarios.

23. **Q: How do you ensure data consistency when integrating multiple systems?**
    - **A:** I use transactions (database transactions, distributed transactions where needed) to ensure ACID properties. For asynchronous integrations, I implement message queues with retry logic and dead-letter queues. I maintain idempotency keys to handle duplicate requests, implement proper error logging and monitoring, and use compensating transactions for rollback scenarios. For ERP integrations, I ensure master data synchronization and maintain audit trails for all data exchanges.

24. **Q: What's your approach to building scalable and maintainable APIs?**
    - **A:** I follow layered architecture separating concerns (Controllers, Services, Repositories), use dependency injection for loose coupling, implement repository and unit of work patterns, apply proper validation and error handling, use DTOs for data transfer, implement caching where appropriate, design for idempotency, version APIs properly, and document thoroughly. I also implement rate limiting, proper logging, and monitoring for production APIs.

25. **Q: How do you handle third-party API failures and timeouts?**
    - **A:** I implement retry logic with exponential backoff for transient failures, use circuit breaker pattern to prevent cascade failures, set appropriate timeouts, implement fallback mechanisms where possible, log all integration failures with context for troubleshooting, and notify appropriate teams for critical failures. For ERP systems, I queue failed requests for manual review or automatic retry, ensuring no data is lost.

Reporting & Business Intelligence

26. **Q: What experience do you have with SQL Server Reporting Services (SSRS)?**
    - **A:** I've built comprehensive SSRS reports including tabular reports, matrix reports, charts, and drill-down reports. I implement parameterized reports for user flexibility, use shared data sources and datasets for reusability, create report subscriptions for automated delivery, and implement role-based security. I optimize report queries for performance and export reports to PDF, Excel, and Word formats for business users.

27. **Q: Describe your experience with SSIS (SQL Server Integration Services).**
    - **A:** I've built ETL packages for data migration, data warehousing, and system integration. I use control flow tasks for workflow orchestration, data flow tasks for transformation (lookups, aggregations, derived columns), implement error handling and logging, schedule packages using SQL Agent, and optimize package performance for large datasets. I've migrated data between

different systems, synchronized data across databases, and automated routine data processing tasks.

Version Control & Team Collaboration

28. **Q: How do you use version control in your projects?**
    - **A:** I use Git and TFS for version control. I follow branching strategies (feature branches for development, main/master for production), commit frequently with meaningful messages, create pull requests for code review, resolve merge conflicts carefully, and tag releases. I understand the importance of clean commit history and use Git rebase when appropriate. I've also worked with SVN in legacy projects.

29. **Q: Describe your experience working in Agile teams.**
    - **A:** I've worked in Agile/Scrum environments throughout my career. I participate actively in sprint planning (breaking down user stories, estimating effort), daily standups (sharing progress and blockers), sprint reviews (demonstrating completed work), and retrospectives (continuous improvement). I collaborate closely with business analysts, product owners, and QA teams. I value iterative development, regular feedback, and adaptive planning.

30. **Q: Why are you interested in this MS Dynamics NAV/Business Central position?**
    - **A:** I'm genuinely excited about this opportunity because it aligns perfectly with my career goals. I have strong ERP development experience and expert-level C#, .NET Framework, and SQL Server skills that directly translate to Dynamics NAV/Business Central. I've built similar business modules and understand enterprise workflows. I'm eager to specialize in the Dynamics ecosystem and learn C/AL and AL. My quick learning ability, strong technical foundation, and passion for ERP systems make me confident I can quickly become productive and contribute meaningfully to your projects. I'm available on short notice and ready to commit long-term to mastering this platform.
    - **A:** I see that you value solid engineering practices like OOP and Design Patterns, which are my core strengths. I'm also excited about the opportunity to work with modern tech like Angular and Cloud platforms. Your focus on innovation [mention something specific about the company if known] really appeals to my desire to solve complex problems in a structured way.