Comparison: BOI FreeDa vs. in-house development with low-code

Criteria	BOI FreeDa	In-house development with low-code
Time-to-value	Can be introduced within a few weeks	Initial applications can be created quickly; however, they often require further work to ensure stability and security
Audit trail & compliance	Out of the box: audit trail, versioning, traceable changes including user and timestamps	Must be modeled manually using low-code building blocks; often limited in terms of audit compliance
Governance & roles	Fine-grained role and rights management	Role and permission management is usually rudimentary or must be implemented individually
Maintenance processes	Standardized 2-, 4-, and 6-eye processes, including change view	Processes can be modeled, but complex validation mechanisms must be implemented individually
Validation logic	Automatic assurance of database compliance; rule-based and verifiable mechanisms directly at the table level	Rules and tests must be defined and tested individually
Excel integration	Seamless import/export workflows	Possible via connectors, but often with limited error handling
Database connection	Supports all major relational databases	Usually limited to specific databases or dependent on connectors
APIs & integration	Standardized interfaces for automated processes	Interface modeling is possible, but is often not versioned or difficult to make backward compatible
UI/UX for operating departments	Optimized for table maintenance without database knowledge	Standard interfaces are often generic, with limited usability for complex tables

Comparison: BOI FreeDa vs. in-house development with low-code

Scalability & performance	Stable architecture for large datasets (>10.000 tables) and many concurrent users (100+)	Depended on platform limitations; complex load scenarios are difficult to test
Security	Proven security mechanisms and comprehensive logging	Security features depend on the low-code platform; often difficult to audit
Support & maintenance	Manufacturer support including regular product updates	Depends on the low-code provider; in-house development required for specific customizations
Total cost of ownership	Plannable: license + implementation + operation	Initially low, but rises sharply due to maintenance, platform dependency, and internal costs
Risk	Low: production-tested, audit-proof, and stable	Medium to high: platform lock-in, limited auditability, custom extensions are risky
Flexibility	High configurability, extensibility via APIs and standardized modules	Significant flexibility in mask and process design, but governance and stability are costly