Vergleich: BOI FreeDa vs. Eigenentwicklung mit Low-Code

Kriterium	BOI FreeDa	Eigenentwicklung mit Low-Code
Time-to-Value	Produktiver Einsatz in wenigen Wochen möglich	Erste Anwendungen schnell erstellt; oft jedoch Nacharbeit nötig für Stabilität und Sicherheit
Revisionssicherheit & Compliance	Out of the box: Audit Trail, Versionierung, nachvollziehbare Änderungen inkl. Benutzer/Zeitstempel	Muss mit Low-Code-Bausteinen selbst modelliert werden, oft eingeschränkt auditfest
Governance & Rollen	Granulare Rollen- und Rechteverwaltung	Rollen- und Rechteverwaltung meist rudimentär oder individuell umzusetzen
Pflege-Prozesse	Standardisierte 2-, 4- und 6-Augen-Prozesse inkl. Änderungsansicht	Prozesse können modelliert werden, aber komplexe Prüfmechanismen müssen individuell abgebildet werden
Validierungslogik	Automatische Sicherstellung der Datenbankkonformität; regel- und prüfbare Mechanismen direkt auf Tabellenebene	Regeln und Prüfungen müssen individuell definiert und getestet werden
MS Excel-Anbindung	Nahtlose Import/Export-Workflows	Über Konnektoren möglich, aber oft mit limitiertem Fehler-Handling
Datenbank-Anbindung	Alle gängigen relationalen Datenbanken werden unterstützt	Meist auf bestimmte Datenbanken limitiert oder durch Konnektoren abhängig
APIs & Integration	Standardisierte, versionierte Schnittstellen für automatisierte Prozesse	Schnittstellenmodellierung möglich, aber oft nicht versioniert oder schwer rückwärtskompatibel
UI/UX für Fachbereiche	Optimiert für Tabellenpflege ohne Datenbankkenntnisse	Standard-Oberflächen oft generisch, eingeschränkte Usability für komplexe Tabellen

Vergleich: BOI FreeDa vs. Eigenentwicklung mit Low-Code

Skalierbarkeit & Performance	Bewährte Architektur für hohe Datenmengen (>10.000 Tabellen) und viele Concurrent User (>100)	Abhängig von Plattform-Limitierungen; komplexe Lastszenarien schwer testbar
Sicherheit	Bewährte Sicherheitsmechanismen und durchgängige Protokollierung	Security-Funktionen abhängig von Low-Code-Plattform; oft schwer auditierbar
Support & Wartung	Herstellersupport inkl. regelmäßige Produkt-Updates	Abhängig vom Low-Code-Anbieter; Eigenaufwand für spezifische Erweiterungen
Total Cost of Ownership	Planbar: Lizenz + Einführung + Betrieb	Anfangs niedrig, steigt aber stark durch Wartung, Plattformabhängigkeit und Eigenaufwand
Risiko	Gering: produktiv erprobt, revisionssicher und stabil	Mittel bis hoch: Plattform-Lock-in, eingeschränkte Auditierbarkeit, individuelle Erweiterungen riskant
Flexibilität	Hohe Konfigurierbarkeit, Erweiterbarkeit über APIs und standardisierte Module	Hohe Freiheit in Masken- und Prozessgestaltung, aber Governance und Stabilität kostenintensiv