12 Fähigkeiten & Vorgehen von Low-Code-Ansätzen

- **Kreative Ideen sofort umsetzen können?**
- **Das Usability Design auf Knopfdruck ändern?**
- **Von heute auf morgen eine Mobile-Version implementieren?**
- **Sich auf das, was wirklich wichtig ist konzentrieren?**
- **Sich nicht großartig mit den Sicherheits-Problematiken auseinandersetzen?**
- **Kein nerviger “Boilerplate Code”, der ja eigentlich gar nichts zu der Funktionalität einer Anwendung beiträgt?**

  → **Geht das? Und wenn ja - wie? Was können Low-Code-Ansätze?**

mendix Platform Solutions Marketplace Developers Partners Resources Try Mendix for f

Platform Overview AI Mendix IDE App Development Cloud Workflow Extensibility Monitoring Integrations Mobile Gove

IDE AI Collaboration Cloud UI/UX Extensibility Data Governance Support

Everything you need to build applications, all in a single platform

nintex Prozessplattform Nintex nutzen

Leistungsstarke Low-Code-Prozessautomatisierung mit K2 Software

Mit der Einfachheit und Leistungsfähigkeit von Nintex K2 Cloud und Nintex K2 Five können Sie die Prozessoptimierung in Ihrem Unternehmen vorantreiben, indem Sie Menschen, Systeme und Daten miteinander verbinden.

appian

FlutterFlow

Build applications faster than ever

Create beautiful UI, generate clean code, and deploy to the app stores or web in one click. Fully extensible with custom code.

Low-Code Definition

- **Softwareentwicklungsansatz, der es Entwickler:innen ermöglicht, Anwendungen mit einem Minimum an manueller Kodierung zu erstellen**
- **Der Großteil des Anwendungsdesigns und -codes wird möglichst wenig manuellem Programmieraufwand erstellt**
- **Ziel:**
  - Produktivität steigern
  - Komplexität reduzieren
- **Weiterentwicklung einer Fourth Generation Language (4GL)**
- **Vereint mehrere Ansätze:**
  - Model Driven Development (MDD)
  - Rapid Application Development (RAD)
  - Automatic Code Generation
  - Visual Programming (VP)

Quellen:
Robert Waszkowski. 2019. Low-code platform for automating business processes in manufacturing. IFAC PapersOnLine 52-10 (2019), 376–381.

Schlüsselmerkmale und Konzepte des Low-Code-Ansatzes

- **Visuelle Entwicklungsmöglichkeit**
- **Drag-and-Drop Funktion**
- **Geringe Programmierkenntnisse erforderlich**
- **Wiederverwendbare Komponenten**
- **Einfache Skalierbarkeit & Anpassung**
- **SCHNELLE Entwicklung**
  - im Sinne von: schneller Fokus auf das Wesentliche legen können
    - mehr in kürzerer Zeit schaffen

High-Code vs. Low-Code

High-Code / Herkömmliche Entwicklung:

- Quellcode als Grundlage
- Großteil des Quellcodes per Hand zu schreiben
- Hohe Komplexität viel Code, der maintained werden muss
- Aufwändiges Umsetzen von Architektur-Patterns

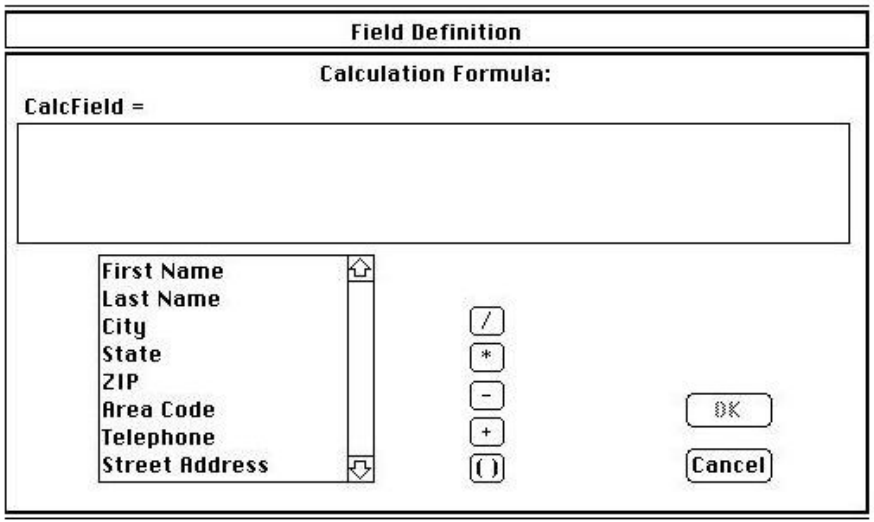
→ WAS brauche ich konkret, und WIE setze ich das um?

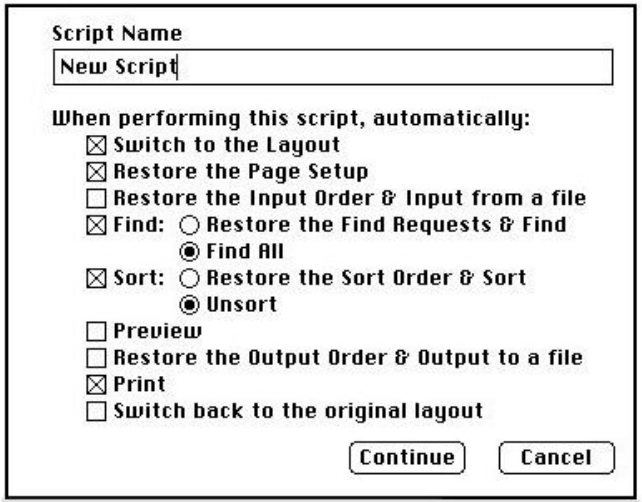
Low-Code:

- Schreiben von Quellcode ist minimal
- Automatische Codegenerierung
- Clean Code als generierter Code
- Einsetzen von vorgefertigten oder selbst geschriebenen Widgets (Code-Segmenten)
- Architektur wird von Anbietern vorgegeben

→ WAS brauche ich?

Low-Code-Ansätze in der Vergangenheit

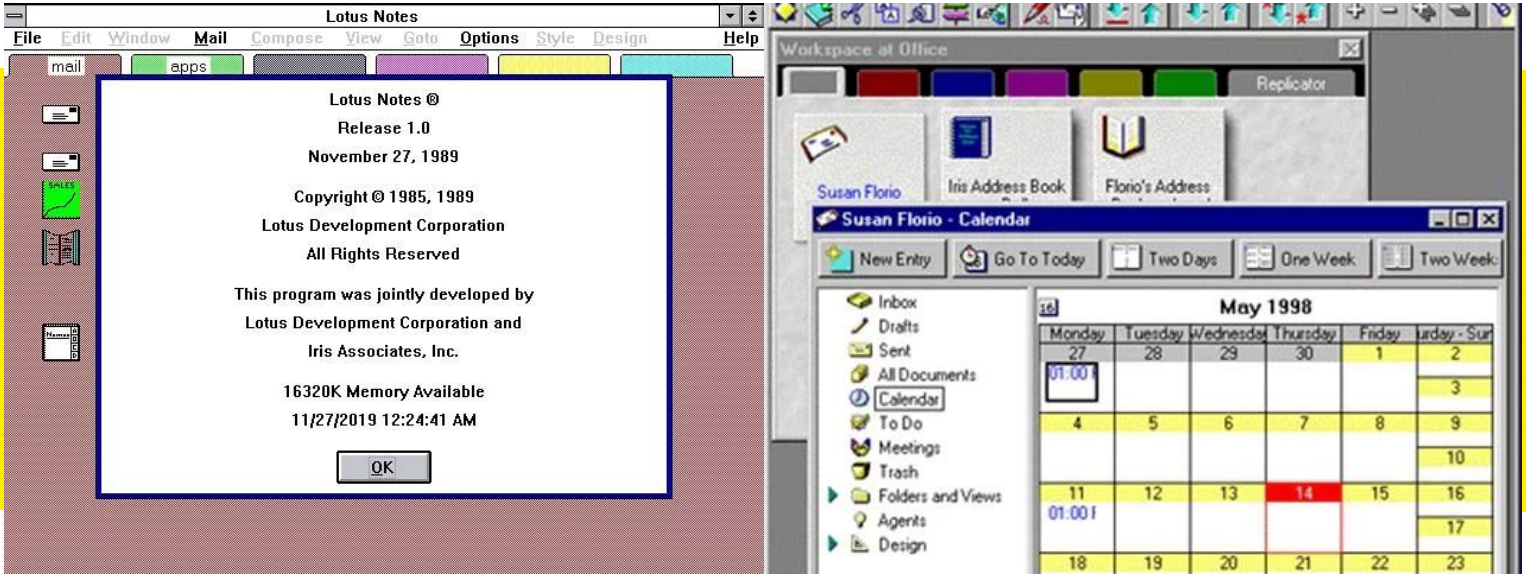




File Maker (1985)

- eigene Skriptsprache

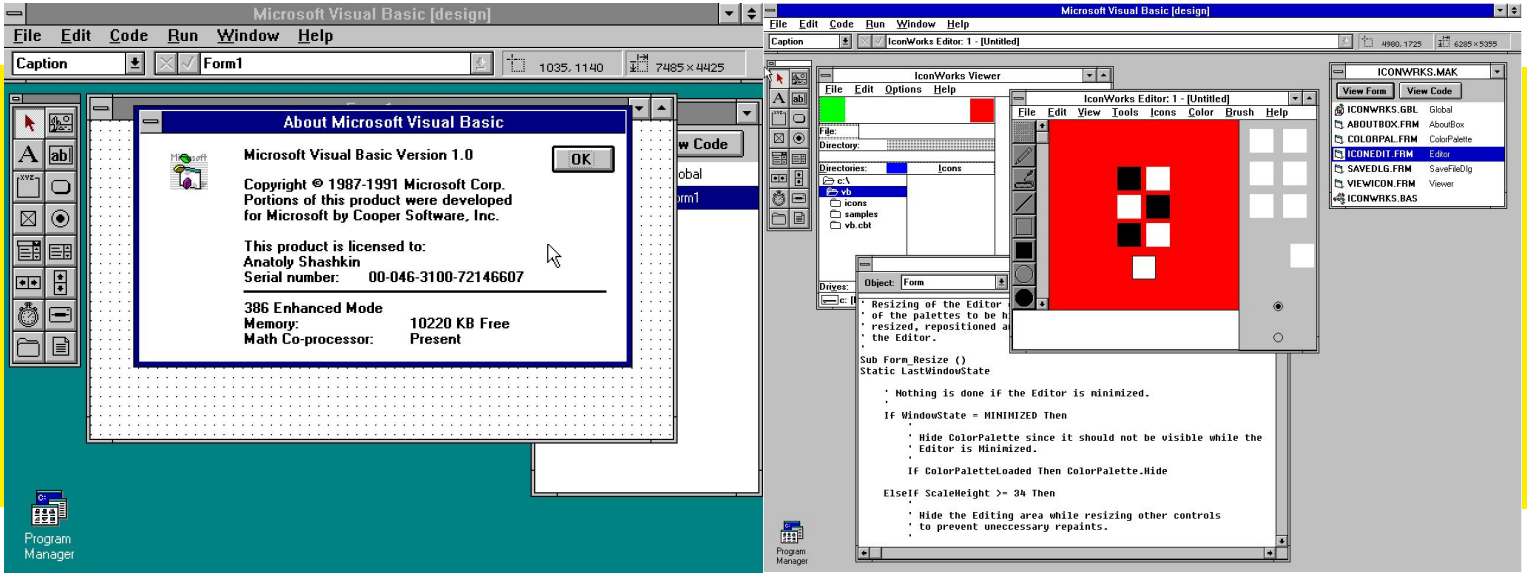
Bildquelle:
https://www.philosophyoffilemaker.com/filemaker-history-64



Lotus Notes (1989)

- einfache Datenbankerstellung mit Formelsprache
- integrierte Tools, z. B. Sicherheitssysteme oder E-Mail

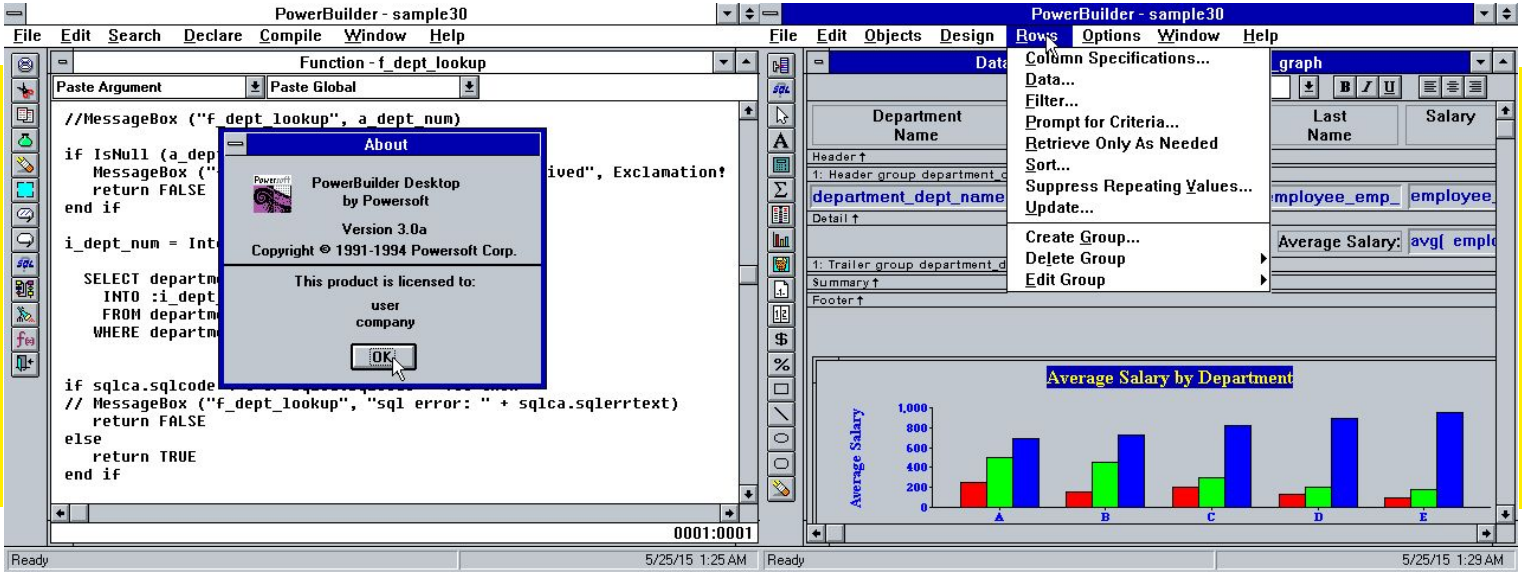
Bildquelle:
https://www.heise.de/news/30-Jahre-Lotus-Notes-Die-Hard-Folge-30-4607567.html



Visual Basic (1991)

- visuelle Benutzerschnittstelle
- drag and drop

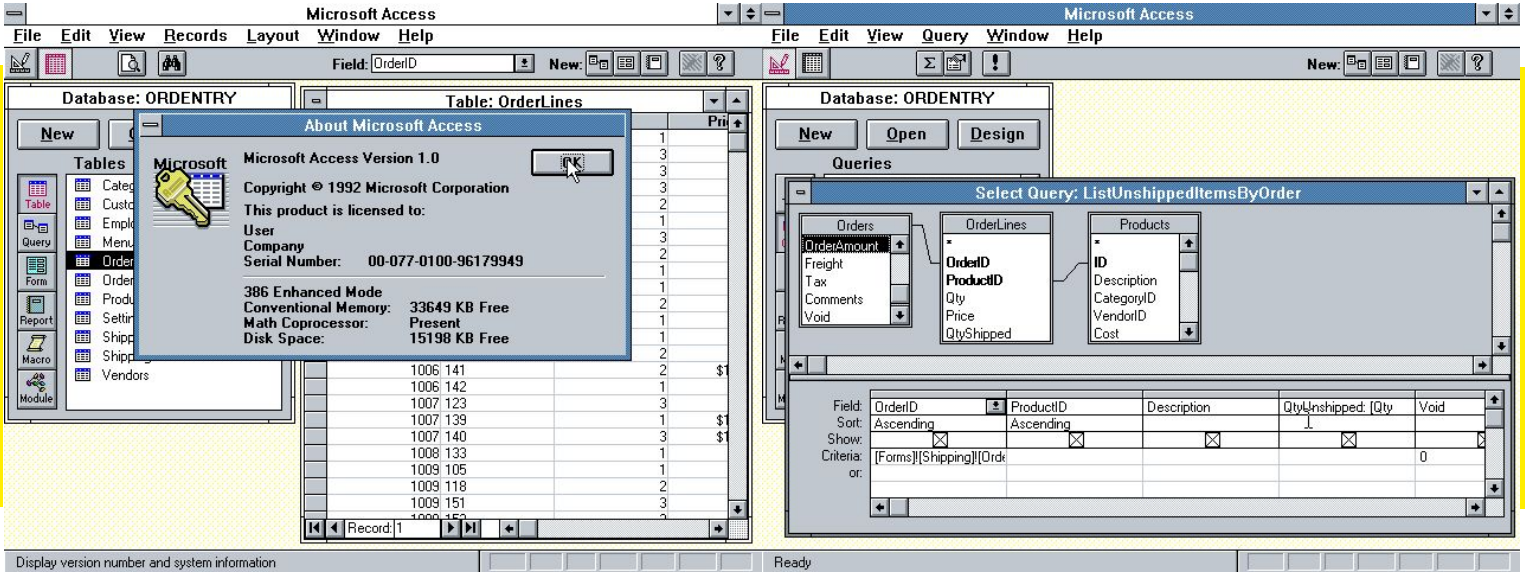
Bildquellen:
https://twitter.com/dosnostalgic/status/846867743752310784
https://socket3.wordpress.com/2016/09/12/visual-basic-early-beginnings/



PowerBuilder (1992)

- automatisch generierter SQL-Code
- DataWindow-Klasse

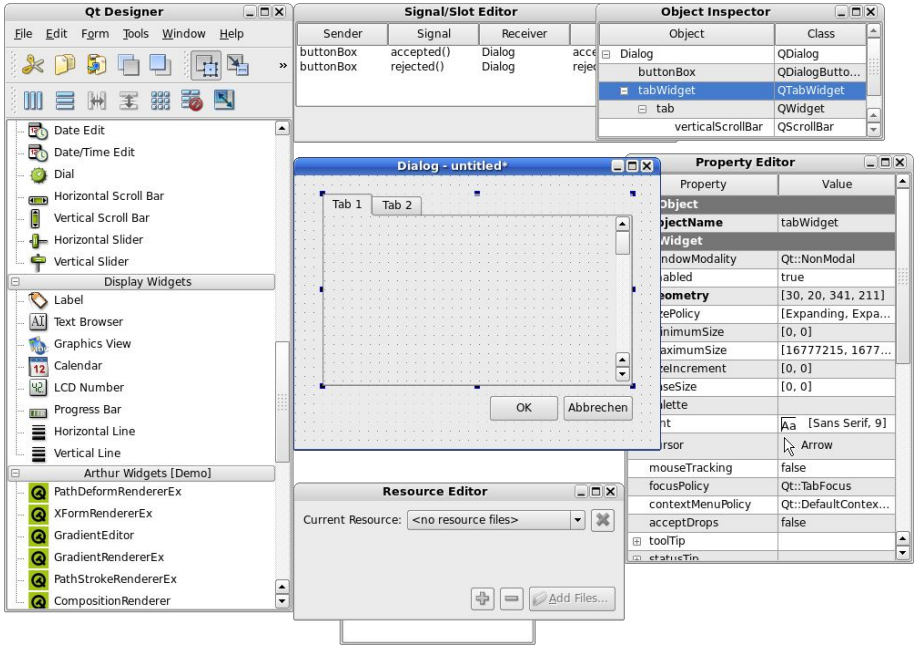
Bildquelle:
https://winworldpc.com/product/powerbuilder/3x



Access (1992)

- Integration von Visual Basic Skripten

Bildquelle:
https://winworldpc.com/product/microsoft-access/10



Qt Designer (1996)

- visuelles Tool zum GUI-Design

Bildquelle:
https://en.wikipedia.org/wiki/File:Qt-designer-v4.2.1.png

Low-Code-Ansätze in der Vergangenheit

- **Abstraktion**
- **Visualisierung**
- **Codegenerierung**
- **Integration eigener Skripte**
- **Integration vorgefertigter Funktionalitäten**

Quelle:
"Dabei ist low-code kein neues Phänomen. Unmengen an kleinen und großen Geschäftsanwendungen sind mit klassischen low-code Werkzeugen wie PowerBuilder, Access, FileMaker oder Lotus Notes erstellt worden und unterstützen – teilweise essentielle – Geschäftsprozesse."
https://www.team4.de/blog/microsoft-power-platform-als-low-code-entwicklungsumgebung/

Moderne Low-Code-Plattformen

Moderne Low-Code-Plattformen

- **Low-Code erlaubt Entwickelnden, sich auf das Wesentliche zu konzentrieren**
- **Angebote beschränken sich nicht mehr nur auf den reinen Entwicklungsprozess**
- **Weitere Dienstleistungen von Low-Code-Anbietern:**
  - Sicherheitsfunktionalitäten
  - Versionsverwaltung
  - Testumgebungen
  - Cloudspeicher
  - Deployment

Citizen Developer

- **Informationstechnische Laien**
- *“developers with little or no software engineering background”* [1]

- **Softwareentwicklung wird zugänglicher**
- *“democratization of technology development beyond IT professionals”* [2]

- **Nachteil: keine Fachkräfte**

Quellen:
[1] Marten Oltrogge et al. “The Rise of the Citizen Developer” (2018)
[2] https://www.gartner.com/en/newsroom/press-releases/2021-06-10-gartner-says-the-majority-of-technology-products-and-services-will-be-built-by-professionals-outside-of-it-by-2024

Das Risiko einer Low-Code-Plattform

Vendor Lock

Abhängigkeit vom Anbieter

- Anbieter löst sich auf
- Fehlende Features
- Vorgefertigte Designs
- Endender Support
- Bei Fehlern seitens der Anbieter kann nicht eingegriffen werden
- Eingeschränkte Flexibilität
  - Anpassungen/Optimierung eventuell aufwändig oder nicht möglich
- Einschränkungen bei der Integration
- Sicherheitsbedenken
  - DSGVO Konformität
- Jeweils Pricing → verschiedene Anbieter mtl. hohe Ausgaben

LIVE DEMO E-SHOP

Low Code in großen Softwarefirmen

Probleme:

- Legacy Code
  - Bugfixing
  - Neue Features
- Dependency Management
  - Kompatibilitätsprobleme
  - Version Conflicts
- Managen von Test Automation und Test Environments
- Scalability
  - Performance muss erhalten bleiben
- Wissensweitergabe & Onboarding
- Security Concerns
- Regulationen (z.B. DSGVO)
- Fachkräftemangel

…

→ **Kommunikation, Koordination & Durchsetzen von Standards**

- Viele Projekte mit verschiedenen Anforderungen
- Teilweise konkurrieren diese untereinander

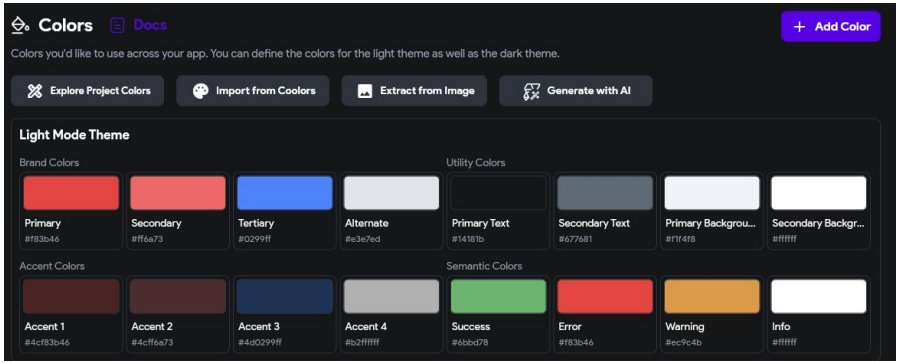
Anwendungsszenarien - Design-Überarbeitung

- **Annahme: kein UI-Design zu Beginn überlegt**
- **Design einer bestehenden App soll überarbeitet werden**

- Anbieter von Low-Code liefern Design-Framework
- Zentrale Anpassung → kein Mühsames Anpassen in jedem Projekt
- Designs sind vorgegeben, in gewissem Rahmen anpassbar

→ Legt Standards fest, alle Änderungen können an SW-Produkte können zentralisiert angepasst werden

Anwendungsszenarien - Design-Überarbeitung



path: Beleg-12-LowCode/lib/flutter_flow/flutter_flow_theme.dart

```dart
class LightModeTheme extends FlutterFlowTheme {
  @Deprecated('Use primary instead')
  Color get primaryColor => primary;
  @Deprecated('Use secondary instead')
  Color get secondaryColor => secondary;
  @Deprecated('Use tertiary instead')
  Color get tertiaryColor => tertiary;

  late Color primary = const Color(0xFF4B39EF);
  late Color secondary = const Color(0xFF39D2C0);
  late Color tertiary = const Color(0xFFEE8B60);
  late Color alternate = const Color(0xFFE0E3E7);
  late Color primaryText = const Color(0xFF14181B);
  late Color secondaryText = const Color(0xFF57636C);
  late Color primaryBackground = const Color(0xFFF1F4F8);
  late Color secondaryBackground = const Color(0xFFFFFFFF);
  late Color accent1 = const Color(0x4C4B39EF);
  late Color accent2 = const Color(0x4D39D2C0);
  late Color accent3 = const Color(0x4DEE8B60);
  late Color accent4 = const Color(0xCCFFFFFF);
  late Color success = const Color(0xFF249689);
  late Color warning = const Color(0xFFF9CF58);
  late Color error = const Color(0xFFFF5963);
  late Color info = const Color(0xFFFFFFFF);
}
```

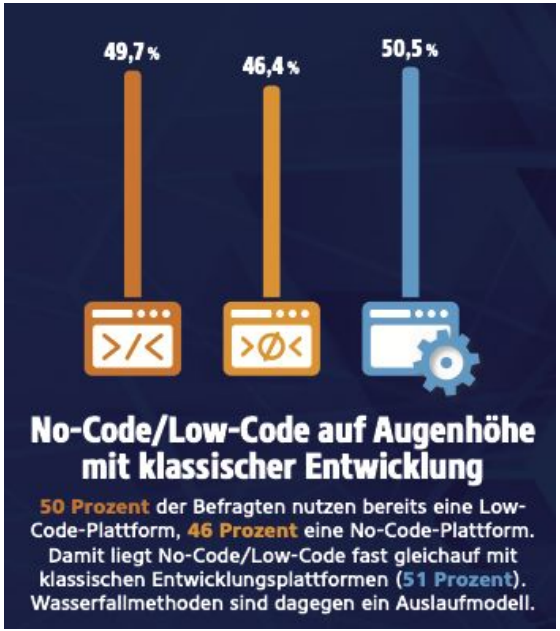
Anwendungsszenarien - Low Code MVPs

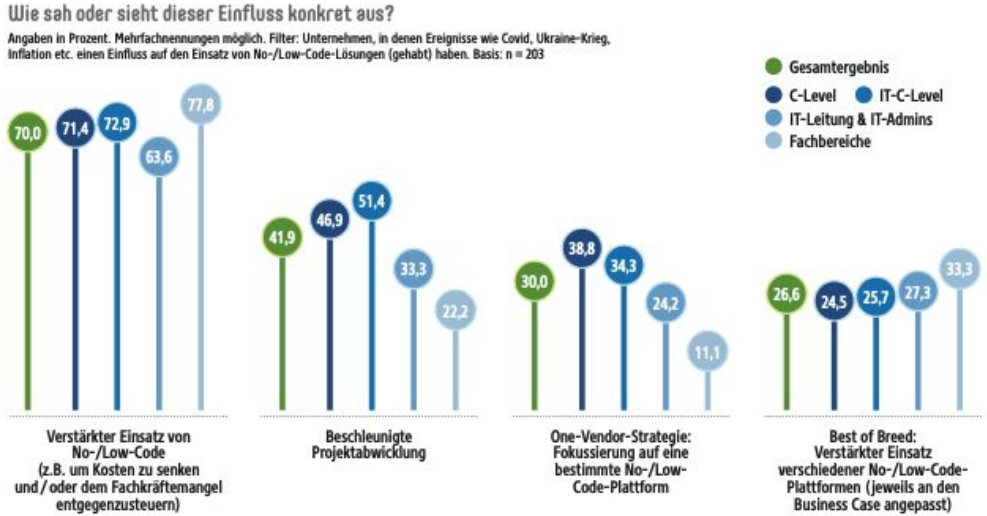
- **Annahme: Low-Code-Entwicklung deutlich schneller als High-Code-Entwicklung**
- **Low-Code-MVPs**
- **Vermeiden des Vendors-Locks durch Weiterentwicklung durch High-Code**

Vorteile & Chancen:

- Höhere Velocity (mehr Requirements umsetzbar)
- MVPs auch für Usertests & Reviews verwendbar → statt separaten Prototypen
- Schnelleres Kundenfeedback
- Mehr Releases möglich

Aktuelle Nutzung von Low-Code





Quelle: No-Code / Low-Code Studie von CIO, CSO und COMPUTERWOCHE Research Services in Zusammenarbeit mit TRANSCONNECT®, München 2023.

Zukunft - wird Low-Code sich weiter durchsetzen?

- Kombination von Low-Code und KI
- Generieren von Source in jeglicher Programmiersprache
  - Könnte High-Code ersetzen
- Trends sind erkennbar - nicht voraussehbar
  - Low-Code könnte verschwinden

Fazit + Diskussion

- im Vorhinein muss man sich darüber Gedanken machen, ob und für welchen Zweck man den Low-Code Ansatz verwenden möchte
  - Langzeitkosten
  - Offene Standards
  - Multi-vendor-Strategie

→ **Es müssen individuelle Entscheidungen getroffen werden**

Dabei zu beachten:
mehr Flexibilität → weniger Komfort
mehr Komfort → weniger Flexibilität

Fragen von und an euch

- Habt ihr schon Erfahrungen mit Low-Code?
- Könnt ihr euch weitere Probleme vorstellen, die beim Einsatz durch Low-Code auftreten können?
- Könnt ihr euch vorstellen, dass High-Code von Low-Code ersetzt wird?

Quellen

Bilder:

Folie 3:

https://www.mendix.com/platform/

https://flutterflow.io/

https://appian.com/de.html

https://www.nintex.de/prozessplattform/k2-software/