



A12

Low Code für individuelle Enterprise Software

2025.06 LTS



Innovation Implemented.

Impressum

Autoren:

Hamarz Mehmanesh
Sabine Gandenberger
Ansgar Weiss
Thomas Kneist
Sebastian Lorenz
Martin Backschat

mgm technology partners GmbH

Taunusstr. 23
80807 München
Tel: +49 (0)89 / 358 680-0

Gerichtsstand und Erfüllungsort: München

*Alle Rechte vorbehalten.
Nachdruck, auch auszugsweise,
nur mit Genehmigung.*

© 2025 mgm technology partners GmbH

www.mgm-tp.com

Inhalt

01	1. Executive Summary	4	04	4. Modellierungsplattform	22
1.1	Was ist A12?	5	4.1	Modellierung als neue Disziplin der Softwareentwicklung	23
1.2	Für welche Einsatzszenarien ist A12 ausgelegt?	6	4.2	Das Modellierungskonzept von A12	23
1.3	Welche Vorteile bringt A12?	7	4.3	Vorteile der »Data first«-Modellierung mit der A12-Regelsprache	31
			4.4	Simple Model Editor (SME)	33
			4.5	Bereitstellung der Modellierungsumgebung	36
02	2. Motivation und Ansatz	8	05	5. Laufzeitplattform	37
2.1	Von Micro-Apps zur integrierten Geschäftsanwendung	9	5.1	Verändertes Aufgabenspektrum für das Entwicklungssteam	38
2.2	Die wahren Aufwände der Enterprise Software Entwicklung	9	5.2	Architektur	40
2.3	Modellgetriebene Entwicklung	14	5.3	Komponenten	34
2.4	Digitale Souveränität und Datenhoheit	15	5.4	Projekt-Template	56
2.5	Künstliche Intelligenz als Katalysator	16	5.5	Betrieb	57
03	3. Plasma UI/UX Design System	18	06	6. Appendix A: Technologien	60
3.1	Effiziente Benutzeroberflächen für Enterprise Software	19			
3.2	Methodik für stimmige User Experience	19			
3.3	Barrierefreie Web-Anwendungen	19			
3.4	Wiederverwendbare Widgets	20			
3.5	Durchdachtes Theming-Konzept	20			



01

Executive Summary

A12 ist eine Plattform für die Entwicklung von Unternehmensanwendungen in komplexen IT-Landschaften. Sie setzt auf modellbasiertes Software Engineering (MDSE) und erschließt das Low Code-Prinzip für die Welt der Enterprise Software.

Dieses Whitepaper stellt den Ansatz von A12 vor und demonstriert, wie Anwendungen damit langfristig zu voll integrierten Unternehmensanwendungen werden können.

1. Executive Summary

Die Struktur der A12 Plattform



Tools, um fachliche Zusammenhänge und Teile der Oberfläche wie Formulare zu gestalten (Low Code)



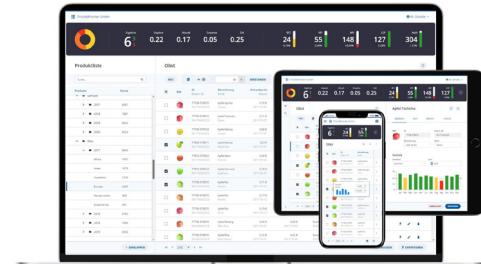
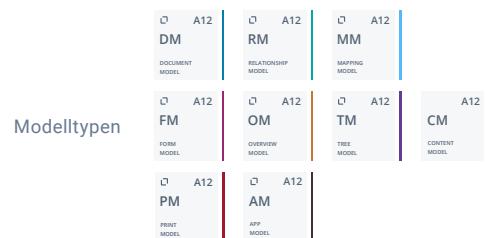
Design-System & Widget-Bibliothek speziell für Geschäftsanwendungen



Client- und Server-seitige technische Basiskomponenten für die Entwicklung robuster und skalierbarer Anwendungen



On Premise und in der Cloud lauffähig



Validierung & Berechnung



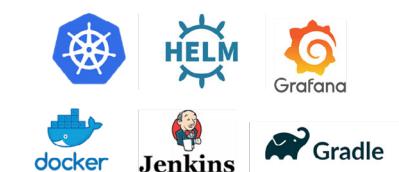
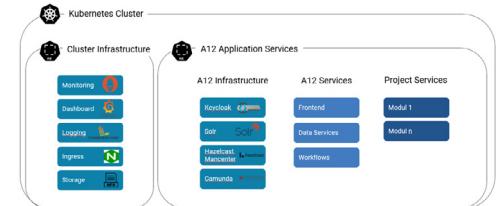
Editor



TypeScript React Redux

Java spring

Kotlin KEYCLOAK



1. Executive Summary

Für welche Einsatzszenarien ist A12 ausgelegt?

Entwicklung komplexer Individualsoftware

A12 ist die ideale Grundlage für hochskalierbare, sichere, robuste und potenziell geschäftskritische Webanwendungen – seien es Antragsmanagement-Systeme, Portale und Fachverfahren für den öffentlichen Sektor, Underwriting-Plattformen für Industrieversicherungen, Kanzleiverwaltungs-Systeme oder eCommerce-Anwendungen.

Ablöse von Excel-/Access-Anwendungen

Trotz all der Nachteile basieren viele Geschäftsabläufe auf Excel- und Access-Lösungen. Mit A12 lassen sie sich konsolidieren und als echte Anwendungen in eine offene Plattform überführen (u.a. mit automatisierten Migrationstools). Dies vereinfacht das Management sowie die Weiterentwicklung und stellt Compliance-Vorgaben sicher.

Basis-Plattform für eine integrierte IT-Landschaft

Gerade mittelständische und große Unternehmen sowie Behörden können massiv von einer Konsolidierung und Modernisierung ihrer IT-Landschaft auf Basis einer einheitlichen Plattform profitieren. A12 ist dafür dank des modularen Aufbaus, robuster Basis-Komponenten und der Ausrichtung auf eine flexible Systemintegration sehr gut geeignet.

Entwicklung von Softwareprodukten

A12 bietet eine erprobte Grundlage, um langlebige Softwareprodukte zu entwickeln. Der Einsatz von A12 verkürzt die Markteinführungszeit und vereinfacht die Entwicklung einer hochwertigen Anwendung, die nicht nur für einen kurzen Moment State-of-the-Art ist, sondern es durch die ständige Weiterentwicklung des Plattform-Unterbaus auch bleibt.



1. Executive Summary

Welche Vorteile bringt A12?



1

Selbstbestimmter Umgang mit fachlichen Inhalten

Fachexperten und Business Analysten können mit Modellierungs werkzeugen den fachlichen Kern der Software eigenständig abbilden und langfristig pflegen.

- Anpassung fachlicher Aspekte ohne Programmierkenntnisse
- Schnelle Umsetzung fachlicher Änderungen
- Flexibles Modellierungskonzept für die Abbildung sehr komplexer Zusammenhänge
- Mächtige Regelsprache für Validierungen und Berechnungen



2

Offene Plattform statt geschlossenes Ökosystem

A12 ist als offenes System ausgelegt, das eine größtmögliche Flexibilität für die Entwicklung sowie die langfristige Pflege und Weiterentwicklung der Software ermöglicht.

- Flexible Nutzung modular aufgebauter Laufzeitkomponenten
- Konsequenter Einsatz von Open Source Technologien
- APIs für individuelle Erweiterungen auf jeder Ebene
- Volle Kontrolle bzgl. Betrieb – On-Premise oder (Private) Cloud-Betrieb
- Kontrollierbarer und verantwortungsvoller Einsatz von KI



3

Zukunftssichere Plattform für langlebige Software

Die konsequente Trennung von Fachlichkeit und Technik ermöglicht es, auch bei Technologiesprüngen den fachlichen Kern beizubehalten.

- Losgelöste Innovierung der Technik durch modellbasierten Ansatz
- »Data First«-Prinzip für nachhaltige fachliche Modellierung
- Sorgfältige Technologie-Auswahl und Nutzung von Industrie-Standards
- Kontinuierliche Weiterentwicklung der technischen Basis

02

Motivation & Ansatz

Das Ziel von mgm besteht darin, schneller und wirtschaftlicher robuste, sichere und langlebige Enterprise Software zu bauen. Dahinter steht der Anspruch erfahrener Software-Ingenieure, die typischen Aufwände der Entwicklung von Geschäftsanwendungen sukzessive zu reduzieren.

Ausgehend von langjährigen Enterprise-Software-Projekten hat mgm deshalb die Entwicklungsplattform A12 entwickelt, die durch modellgetriebene Abstraktionen eine Trennung von Fachlichkeit und Technik ermöglicht.

2. Motivation und Ansatz

Von Micro-Apps zur integrierten Geschäftsanwendung

Viele Geschäftsanwendungen haben ihren Ursprung in pragmatischen Behelfslösungen einzelner Fachabteilungen. Der Klassiker sind Excel-Tabellen, die klein starten und durch die sukzessive Erweiterung und den Einsatz von Makros bereits einen Anwendungscharakter erhalten. Nachteil dieser pragmatischen Vorgehensweise (»Schatzen-IT«) sind große Risiken mit Blick auf Datenschutz und IT-Sicherheit.

Low Code Plattformen zielen darauf, diesen Bruch von der Behelfslösung zu einer Anwendung aufzuheben. Sie er-

möglichen es der Fachabteilung, echte Anwendungen zu bauen – unter Berücksichtigung der firmenspezifischen IT-Richtlinien. Dieser Schritt bietet sich für eine Teilmenge der Behelfslösungen an, die Potential versprechen.

Für eine weitere Teilmenge der Mikroanwendungen – in der Regel genau die geschäftskritischsten Anwendungen – entsteht eine weitere Herausforderung: Sie müssen in eine heterogene IT-Landschaft integriert werden. Die meisten Low Code Plattformen bringen zwar vorgefertigte Lösungen für typische Integrationsszenarien mit. Sie sto-

ßen aber an Grenzen. Individuelle Entwicklungen und eine professionelle Systemintegration sind unvermeidbar.

A12 adressiert nicht nur den Übergang zu Mikroanwendungen, sondern auch die Evolution zu integrierten Geschäftsanwendungen. Gerade hier treten langfristig die größten Aufwände in der Entwicklung, der Wartung und dem Betrieb von Geschäftsanwendungen auf.

Die wahren Aufwände der Enterprise Software Entwicklung

Als Softwarehaus entwickelt mgm seit über 25 Jahren individuelle Enterprise Software. Der Kerngedanke von A12 geht auf eine Reihe wiederkehrender Beobachtungen aus den unterschiedlichsten Projekten zurück. Vor allem die typischen Aufwandstreiber, die erst im Laufe eines Projekts auftreten – allen voran fachliche Anpassungen und Integrationen – werden immer wieder unterschätzt und sind langfristig für das Scheitern auch sehr großer IT-Projekte verantwortlich.

Geschäftsdomänen – alles andere als Standard

Jede Enterprise Software modelliert einen bestimmten Ausschnitt der unternehmensspezifischen Realität. Grundlage des jeweiligen Modells ist die Geschäftsdomäne. Die Geschäftsdomäne besteht aus einer Reihe von (Geschäfts-) Entitäten. Das können beispielsweise Kunden, Produkte und Bestellungen sein. Jede dieser Entitäten ist in der Software durch ein Entitäten-Modell repräsentiert. Es definiert die Struktur der Entität sowie seine Attribute und Beziehungen zu anderen Entitäten.

Die Modelle der Entitäten unterliegen einem kontinuierlichen Wandel. Das ist ein ganz wesentlicher Aufwandstreiber der Enterprise Software Entwicklung.

Verantwortlich für den Wandel sind vor allem folgende Punkte:

- ⊕ Ein gewachsenes Geschäft ist ein komplexes Geflecht. Das Wissen darüber ist über viele Köpfe verteilt. Es gibt viele Stellschrauben, an denen ganz unterschiedliche Unternehmensvertreter immer wieder drehen.
- ⊕ Das Geschäft entwickelt sich kontinuierlich weiter. Das Portfolio ändert sich. Vertriebswege treten hinzu oder fallen weg. Je nach Branche müssen neue regulatorische Vorgaben eingehalten werden.
- ⊕ Unternehmen organisieren ihr Geschäft individuell nach ganz unterschiedlichen Regeln. Darüber hinaus verwenden sie eigene Terminologien, die sukzessive erweitert und angepasst werden.

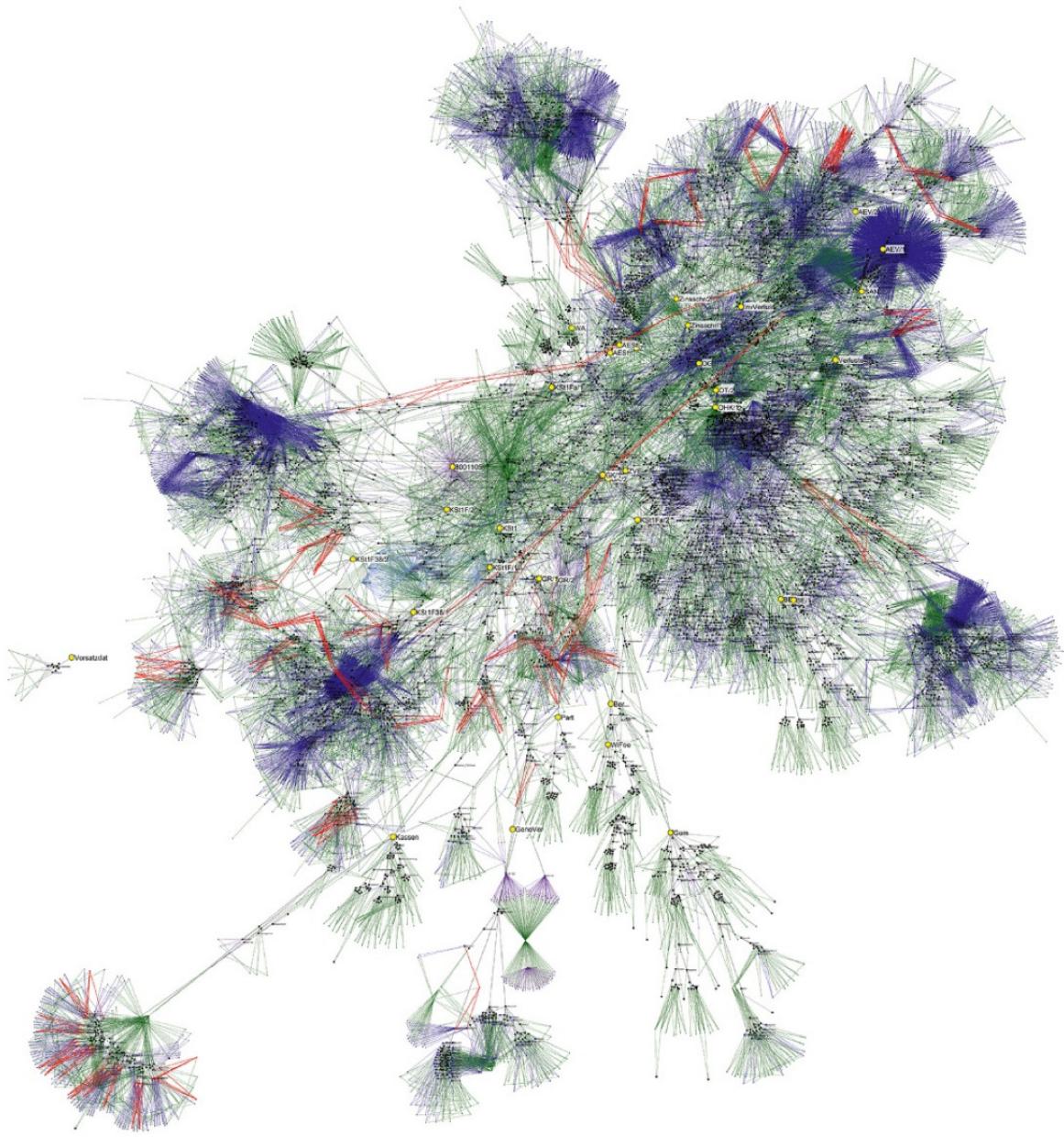


2. Motivation und Ansatz

Die Modelle, die jeder Enterprise Software zugrunde liegen, folgen dementsprechend keinem Standard. Im Gegenteil: Sie sind hochindividuell und enthalten immer wieder Ausnahmen und in einigen Fällen auch fachliche Inkonsistenzen.

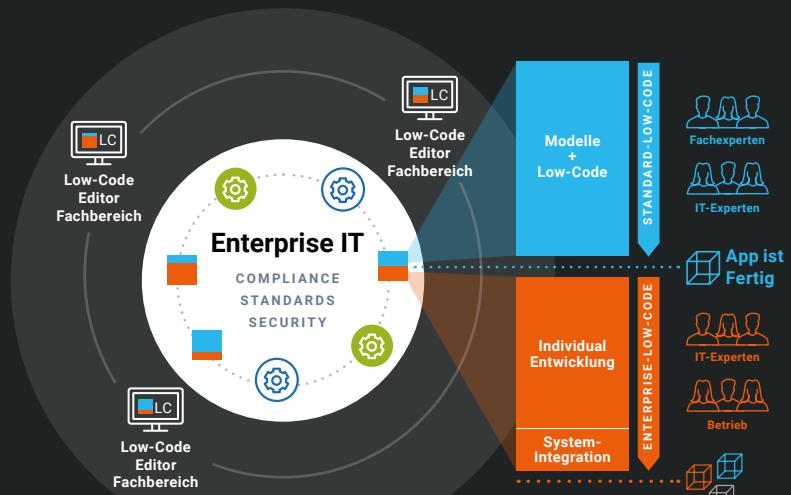
Ein weiterer Aufwandstreiber hängt ebenfalls mit den Modellen zusammen: Die Modelle werden sukzessive komplexer. Sie bilden immer größere Ausschnitte der Realität ab, die für den Erfolg in der jeweiligen Geschäftsdomäne relevant sind.

Die Abbildung demonstriert, wie hoch beispielsweise die Komplexität bei der Abbildung eines Steuerformulars werden kann.



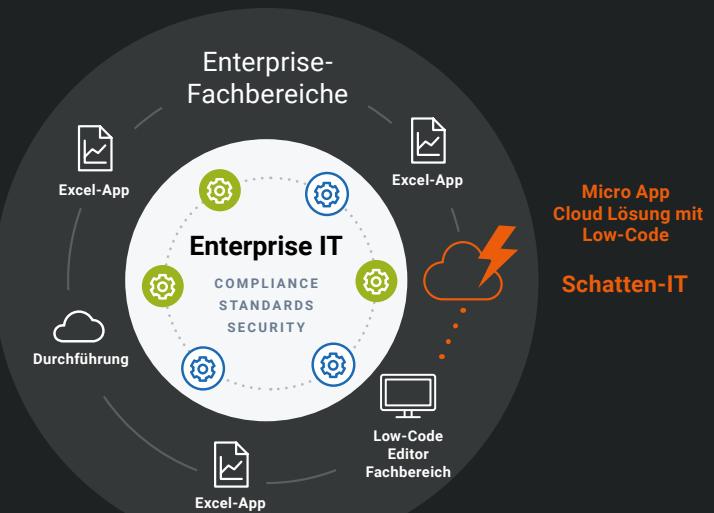
Enterprise-IT und Schatten-IT

Die klassische Enterprise-IT von Unternehmen ist zentral aufgebaut und besteht aus Standardsoftware und einer Vielzahl von individuell entwickelter Software. Doch Anforderungen in Fachbereichen werden zunehmend außerhalb der Enterprise-IT gelöst. Sie entstehen z. B. als Excel-Lösungen oder als einfache Micro-Apps. Oft geschieht das ohne Schnittstellen zur IT. Schatten-IT entsteht.



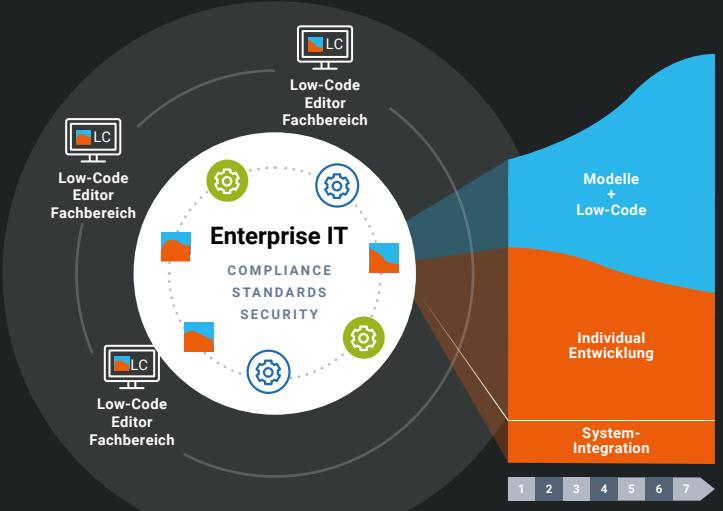
Dynamische Gewichtung von Low Code- und Individualentwicklung

Die Gewichtung von Low Code- und Individualentwicklung kann nicht nur von Projekt zu Projekt sehr unterschiedlich sein. Auch innerhalb der Lebensphasen eines IT-Projektes schwanken die Aufwände im Bereich der Modellierung sowie Individualentwicklung und Systemintegration.



Enterprise Low Code Development

Wenn der Fachbereich mit den passenden Low Code Werkzeugen Anwendungen selbst gestaltet und die IT gleichzeitig zentrale Technologie und Standards sichern kann, entstehen wirklich wertvolle Geschäftsanwendungen. Je nach Projekt kann der Low Code Anteil individuell gewichtet werden. Ziel ist, dass Fachbereich und IT Experten wirkungsvoll zusammenarbeiten.



2. Motivation und Ansatz

Unterschiedliche Repräsentationen von Geschäftsentitäten

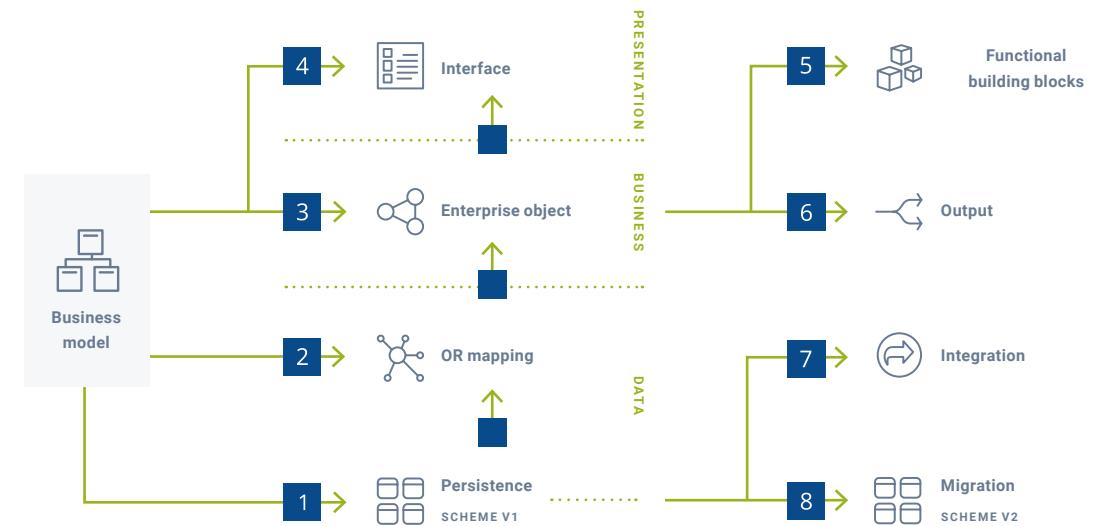
Aus Perspektive der Softwareentwicklung sind jegliche Änderungen der Geschäftsentitäten mit hohem Aufwand verbunden. Der Grund: Sie müssen in unterschiedlichen technischen Zusammenhängen unterschiedlich repräsentiert werden. Zwischen diesen unterschiedlichen Repräsentationen sind Abbildungen nötig.

Die Repräsentation müssen aufeinander gemappt werden. In der Praxis bringt dadurch selbst die kleinste fachliche Änderung eine ganze Kaskade von Anpassungsschritten in der Software mit sich. Die Abbildung zeigt eine Übersicht dieser verschiedenen Repräsentationen.

Den Ausgangspunkt bildet das **Fachmodell**. Es stellt als Vorlage für die Implementierung die erste Repräsentation der modellierten Geschäftsentitäten und ihrer Beziehungen zueinander dar.

In einer Drei-Schichten-Architektur treten folgende Repräsentationen hinzu:

→ In der **Datenhaltungsschicht** sind die Geschäftsobjekte in einer Datenbank abgelegt. Hierfür ist eine Repräsentation erforderlich, die den Vorgaben dieser Persistenzebene genügt (1). Bei einer relationalen Datenbank nimmt sie eine tabellarische Form an. Um die Daten der Tabellen in einer objektorientierten Hochsprache wie Java verarbeitbar zu machen, findet eine Objektrelationale Abbildung statt (2).



→ In der **Businessschicht** befindet sich die Anwendungslogik. Sie bringt nochmal eine eigene Repräsentation (3) mit sich, die sich aus der jeweiligen Verarbeitung der Geschäftsobjekte, -komponenten und Workflows ergibt.

→ In der **Präsentationsschicht** ist wieder eine andere Repräsentation (4) erforderlich. Hier geht es darum, wie die Geschäftsentitäten in der Benutzeroberfläche dargestellt werden und welche Interaktionen mit ihnen möglich sind.

Darüber hinaus fallen weitere Repräsentationen und Abbildungen der Geschäftsentitäten in folgenden Zusammenhängen an:

- Bereitstellung von Services für bestimmte Funktionalitäten – zum Beispiel das Abrufen des Warenbestands (5).
- Generierung von Word- oder PDF-Dokumenten beispielsweise Versicherungspolicen oder Bescheide (6).
- Integration mit weiteren Systemen innerhalb der IT-Infrastruktur des Unternehmens (7).
- Umfangreiche Migrationen, die aufgrund von Weiterentwicklungen des Schemas der zugrundeliegenden Datenbank erforderlich werden (8).

2. Motivation und Ansatz



Jede dieser Abbildungen bringt ihre eigenen Herausforderungen und Aufwände mit sich. Einige von ihnen treten erst nach einiger Zeit in den Fokus.

So adressiert die gezeigte Abbildung zum Beispiel die Tatsache, dass Geschäftsanwendungen in der Regel nicht für sich alleine stehen. Und wenn sie das tun, dann nicht für lange Zeit.

Sie sind vielmehr eingebettet in komplexe IT-Landschaften und entfalten erst ihr volles Potenzial, wenn sie mit einer Reihe interner und externer Anwendungen verknüpft sind.



Heterogene IT-Landschaften

Die IT-Landschaften von mittelgroßen bis großen Unternehmen teilen branchenübergreifend ein charakteristisches Merkmal. Sie bestehen aus großen Anwendungen wie SAP oder größeren Individualanwendungen sowie einer Mehrzahl von kleineren Anwendungen. In Hinblick auf ihre Einbindung in der Abwicklung von Geschäftsvorfällen und den Datenaustausch arbeiten alle Anwendungen im Idealfall integriert. Aufgrund der Vielfalt der eingesetzten Technologien (SAP, Java-Anwendungen, Cloud-basierte Anwendungen, etc.) sind diese IT-Landschaften in der Regel auf einer **dauerhaft heterogenen technologischen Basis** gebaut. Das bedeutet im Einzelnen:

- Jede Anwendung in dieser Landschaft hat in der Regel ein eigenes Projektteam, eigene Release Zyklen und eine eigene technologische Basis
- Bei Individualanwendungen werden je nach Alter der Anwendung und den Vorlieben und Entscheidungen des Projektteams jeweils andere Technologie- und Architekturentscheidungen getroffen. Dies gilt auch für eingebundene Anwendungen, die auf einem Produkt wie SAP oder MS Dynamics bestehen.
- Unterschiedliche Anwendungen haben in der Regel auch unterschiedliche Ansprechpartner auf der Fachseite, die die fachlichen Vorgaben für die Erst- und Weiterentwicklung der Anwendung erstellen.

Die Heterogenität der IT-Landschaft ist ein weiterer Aufwandstreiber, der sich auch durch die heutigen Low Code Ansätze nicht vollständig auflösen lässt. Individuelle Entwicklungsarbeiten lassen sich reduzieren, bleiben aber nicht vollständig aus. Auch wenn Geschäftsanwendungen klein starten, entstehen meist früher als später Integrationsfragen – denn erst integriert entfalten die Anwendungen ihr volles Potenzial.

2. Motivation und Ansatz

Modellgetriebene Entwicklung

Wesentliche Aufwandstreiber der Enterprise Softwareentwicklung sind die vielen unterschiedlichen Repräsentationen von Geschäftsentitäten in unterschiedlichen Schichten der Software. Wie lassen sich die Aufwände der Abbildungen zwischen diesen Repräsentationen reduzieren? Eine Antwort liefert die modellgetriebene Softwareentwicklung. Die Idee dahinter: Geschäftsentitäten und ihre Beziehungen untereinander werden in Modellen abgebildet. Mit Hilfe spezieller Editierwerkzeuge lassen sich die Modelle definieren und anpassen.

Durch spezielle Interpreter und Code-Generatoren werden die Modelle in die Anwendung gebracht. Der Clou: Die Abbildungen der unterschiedlichen Repräsentationen müssen nicht mehr händisch umgesetzt werden. Das erledigen die Generatoren und Interpreter. Damit können fachliche Inhalte, die in Enterprise Software wie oben beschrieben einem ständigen Wandel unterliegen, deutlich schneller und mit geringerem Aufwand in der Software abgebildet werden.

Vorteile modellbasierter Entwicklung



- ⌚ **Termingerechte Umsetzung**
Modellbasierte Entwicklung ermöglicht eine termingerechte Umsetzung und Auslieferung der IT Systeme, auch für sich häufig ändernde Fachvorgaben.
- ⇄ **Vereinfachtes Abhängigkeitsmanagement**
Eine modellbasierte Gesamtarchitektur vereinfacht das Abhängigkeitsmanagement. Sie erlaubt die Trennung von Fachlichkeit und technischem Rahmen in getrennte Releasezyklen. Die Fachlichkeit kann außerdem für jede Version und jede Datenart in spezifische Modelle zerlegt werden, die jeweils explizit versioniert, aber pro Version und Datenart unabhängig sind. Auch fachliche Änderungen können bei Bedarf pro Datenart und Version unabhängig voneinander produktiv geschaltet werden.
- 🔧 **Verringelter Testaufwand**
Für individuell erstellte, sich häufig ändernde Software ist der Testaufwand extrem hoch. Jede Version, jede Datenart und jede Änderung muss für jedes Produkt separat getestet werden. Bei modellbasierter Entwicklung beschränkt sich der fachliche Testaufwand hingegen auf die Modelle.
- 👓 **Freier Weg für technische Innovationen**
Durch die Trennung von Fachlichkeit und Technik können technische Innovationen realisiert werden, ohne sämtliche fachlichen Inhalte der Anwendung berücksichtigen zu müssen. So lässt sich zum Beispiel eine neue Technik in der Gestaltung und Realisierung der Benutzeroberfläche, in der Persistierung oder in der Server-Verarbeitung einführen.

2. Motivation und Ansatz

Digitale Souveränität und Datenhoheit

Als langjähriger Partner öffentlicher Verwaltungen teilen wir die Werte unserer Auftraggeber für einen selbstbestimmten Umgang mit Software. A12 befähigt Fachabteilungen dazu, selbst bei hochkomplexen, integrierten Anwendungen die volle Kontrolle zu behalten.

Kontrolle über Fachlichkeit - Austauschbarkeit von Technik

Die strikte Trennung von Fachlichkeit und Technik schafft große Flexibilität bei der langfristigen Weiterentwicklung der Software. Der Kern der Software liegt in den fachlichen Modellen, die auch ohne Softwareentwickler angepasst und weiterentwickelt werden können. Sie liegen in einem offenen Format als einfache JSON-Dateien vor.

Die Technik wird durch diese Isolierung fachlicher Inhalte viel einfacher austauschbar, als wenn fachliche Aspekte eng in den Code verwoben wären. Das sichert insbesondere, dass nicht bei jeder technischen Neuerung alles neu entwickelt werden muss. Die Technik lässt sich viel einfacher auf einem aktuellen Stand halten.

Kontrolle über Daten

Insbesondere bei geschäftskritischer Software ist es unerlässlich, dass sensible Daten in einer vertrauenswürdigen, sicheren Umgebung liegen und der reibungslose Betrieb sichergestellt ist. Wie wichtig es ist, die volle Kontrolle über den Betrieb zu behalten, sehen wir beispielsweise immer wieder bei unseren Kunden aus dem E-Commerce-Sektor. Die Systeme laufen zu Zeiten hoher Anfragen wie im Weihnachtsgeschäft für lange Zeit auf Höchstlast ohne Downtime. Dafür muss die Software zum Einen hochperformant und skalierbar sein. Zum Anderen ist aber auch die alleinige Kontrolle über die zugrunde liegende Infrastruktur und die verwendeten Release-Stände nötig.

Für die Bereitstellung von A12 bieten wir folgende Optionen an:

- On-Premise Betrieb im unternehmenseigenen oder externen Rechenzentrum**
- Betrieb in der Private Cloud von mgm, gehostet in einem Rechenzentren in Deutschland**
- Cloud-Betrieb bei einem beliebigen Cloud-Anbieter**

2. Motivation und Ansatz

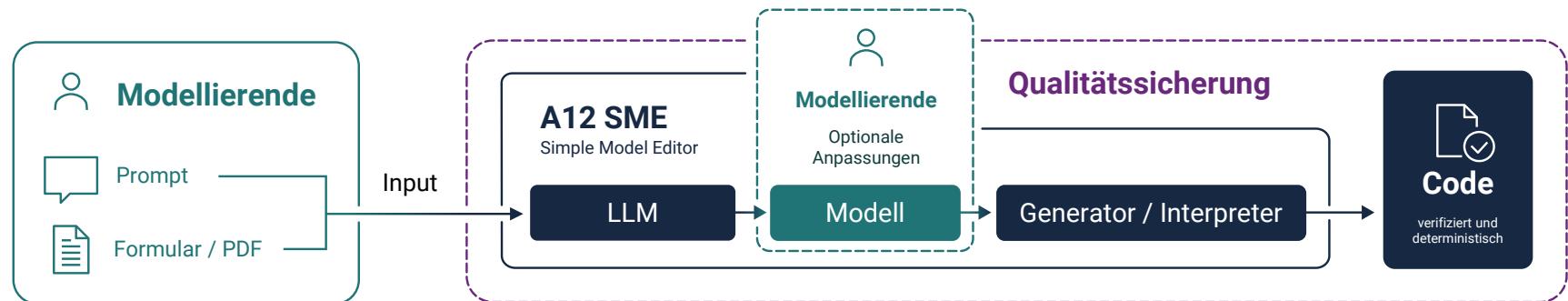
Künstliche Intelligenz als Katalysator

KI-basierte Entwicklungsansätze wie Vibe Coding revolutionieren den Softwareentwicklungsprozess. Sie sind aufgrund der bestehenden Schwächen von LLMs wie Halluzinationen aber (noch) risikobehaftet und im Enterprise-Kontext nur unter hohen Auflagen einsetzbar. Auf Basis von A12 lassen sich viele Fähigkeiten von KI jedoch schon heute für die Enterprise-Entwicklung erschließen. Neben robusten Komponenten und klar strukturierten A12-Modellen hat mgm für A12 eine umfassende Qualitätssicherungslandschaft sowie Tools für Security-Prüfungen und Build & Deployment entwickelt. Damit ist ein kontrollierbarer Rahmen gegeben, in dem sich generative KI und agentenbasierte Systeme sehr effektiv und verantwortungsbewusst einsetzen lassen.

Einsatz von KI auf der Modell-Ebene

Der modellbasierte Entwicklungsansatz von A12 bietet in Kombination mit KI eine Reihe entscheidender Vorteile im Vergleich zu konventionellen Entwicklungsparadigmen. So lassen sich mittels KI die Modelle automatisiert generieren. Dies reduziert manuelle Aufwände und beschleunigt die Umsetzung von Fachanforderungen. Für A12 Dokumentenmodelle, die das Herz der Datenmodellierung darstellen, ist diese Möglichkeit als Prototyp bereits implementiert. Im Simple Model Editor können Modellierer ein Ausgangsdokument und ein Prompt angeben, auf deren Basis automatisch ein initiales Modell erstellt wird, das sich bei Bedarf manuell verfeinern lässt.

Der besondere Vorteil des Einsatzes von KI auf der Modell-Ebene besteht darin, dass weiterhin die reguläre Verarbeitungskette der Modelle in Form von Interpretern und Generatoren zum Einsatz kommt. Der resultierende Code bleibt deterministisch. Darüber hinaus lassen sich auch auf generierte Modelle eine Reihe vorhandener Test- und Qualitätssicherungsverfahren anwenden – z.B. die im Simple Model Editor (SME) integrierte Konsistenzprüfung für Validierungs- und Berechnungsregeln.



2. Motivation und Ansatz

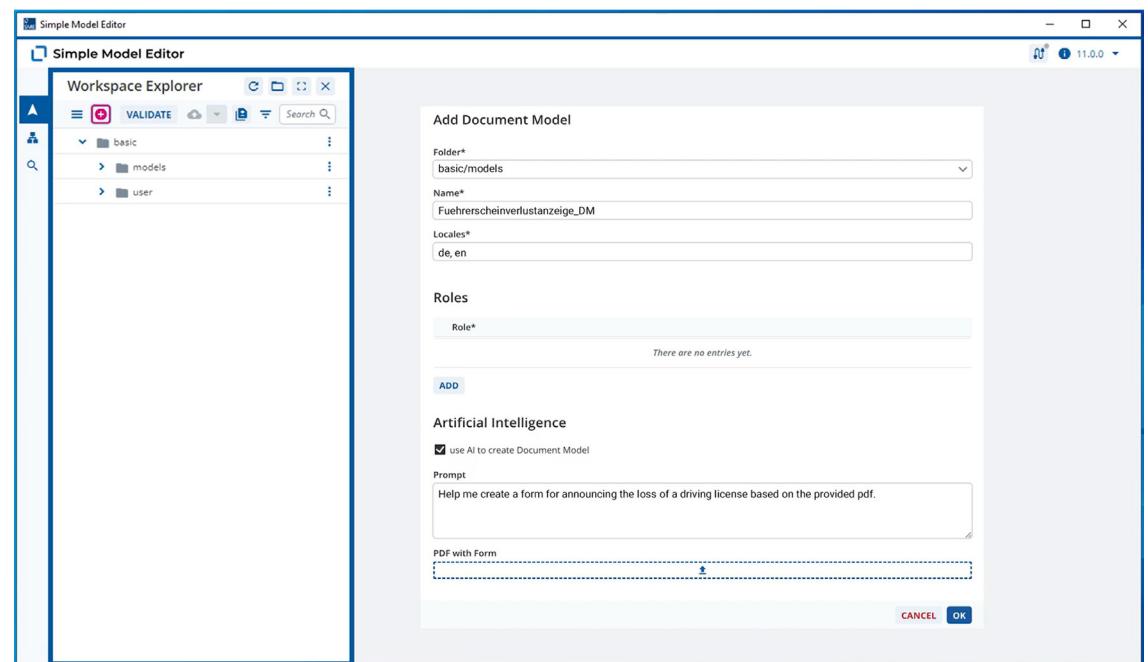
Einsatz von KI auf der Code-Ebene

Auch beim Einsatz von KI auf der Code-Ebene ergeben sich Vorteile für A12-Projekte. Die Komponenten der Laufzeitplattform bieten stabile und kontinuierlich getestete Lösungen für viele typische Herausforderungen von Geschäftsanwendungen – von der Authentifizierung und Autorisierung bis hin zu barrierefreien Widgets. Der Einsatz dieser Komponenten reduziert das Risiko von Sicherheitslücken und erhöht die Produktivität von KI-Bots. Eine umfangreiche Dokumentation der Komponenten und Vorgaben wie Entwicklungsrichtlinien und Security-Guidelines helfen dabei, die Einhaltung von Best Practices sicherzustellen.

Nichtsdestotrotz sind auch beim Einsatz von KI auf der Code-Ebene in A12-Projekten weitere Sicherheitsmaßnahmen unabdingbar. Dazu gehören vor allem Code-Reviews. Generell sollte keine Zeile von KI-generiertem Code ohne menschliche Überprüfung ins Projekt einfließen. Die speziell auf A12 ausgelegten Test- und Sicherheitstools bilden jedoch ein zusätzliches Sicherheitsnetz und helfen dabei, die Risiken von KI in der Softwareentwicklung zu minimieren.

LLM-Ergebnisse messen und kontrollieren

Im Rahmen von KI Use Cases stellt sich immer wieder die Frage, welches Modell die besten Ergebnisse bietet und wie gut die LLM-Ergebnisse ausfallen. mgm hat eine Reihe von Verfahren und Schemata entwickelt, um die Ergebnisse von LLMs kontinuierlich zu messen und zu bewerten. Sie sind in A12-Projekten ein wichtiger Baustein, um KI-getriebene Funktionalitäten mit klar definierten Quality Gates zu implementieren.



03

Plasma UI/UX Design-System

Enterprise Anwendungen sind durch eine hohe Informationsdichte und große Komplexität gekennzeichnet. Designsprachen wie Material Design stoßen da schnell an ihre Grenzen. Sie bieten nur unzureichende Antworten darauf, wie beispielsweise komplexe Tabellen übersichtlich dargestellt werden können – oder wie die Struktur der Benutzeroberfläche konsistent weiterentwickelt wird, wenn neue Informationen hinzukommen. Aus diesem Grund hat mgm das Designsystem A12 Plasma entwickelt.

3. Plasma UI/UX Design-System

Effiziente Benutzeroberflächen für Enterprise Software

Plasma besteht aus einer Reihe von UI/UX-Komponenten, Verwendungsmustern und Gestaltungsrichtlinien, mit denen sich konsistente, effiziente und ansprechende Benutzeroberflächen gestalten lassen. Dabei adressiert Plasma zwei Kernanforderungen der Benutzeroberflächen von Geschäftsanwendungen: Skalierbarkeit und Bewältigung der Komplexität.

Eine zentrale Idee von Plasma besteht darin, die dargestellte Informationsdichte so weit wie möglich zu reduzieren. Die Nutzer bekommen idealerweise nur das präsentiert, was sie für ihre derzeitigen Aufgaben auch wirklich benötigen. Sie können die Aufgaben dadurch schneller und effizienter ausführen.

Methodik für stimmige User Experience

Plasma enthält eine Reihe von wiederverwendbaren Mustern und Komponenten für Anforderungen, die in den Benutzeroberflächen von Geschäftsanwendungen immer wieder auftreten – vom Login-Screen bis zur Validierung von Nutzereingaben. Dazu gehören Muster für den Anwendungsrahmen, Navigationselemente und Benachrichtigungen. Dazu gehören aber auch Konzepte für den Umgang mit Geschäftsobjekten und den typischen Workflows, in die sie eingebunden sind. Plasma ist damit deutlich mehr als eine reine Designsprache. Als Designsystem von A12 bietet es eine umfangreiche Methodik für die Gestaltung stimmiger Nutzererlebnisse in Enterprise Anwendungen.

Barrierefreie Web-Anwendungen

Barrierefreiheit wird für Web-Anwendungen immer wichtiger. Bereits seit einigen Jahren sind öffentliche Stellen innerhalb der EU dazu verpflichtet, Websites und mobile Anwendungen barrierefrei zu gestalten. Ab 2025 müssen laut dem European Accessibility Act bis auf wenige Ausnahmen alle Websites und Web-Anwendungen barrierefrei sein. Mit Plasma ist die A12-Plattform explizit für die Erstellung barrierefreier Web-Anwendungen ausgelegt. Zahlreiche UI-Komponenten – u.a. auch die modellgetriebenen Engines für Formulare und Übersichtslisten – sind barrierefrei out-of-the-box. In der Projektpraxis von individueller Software sind jedoch auch immer zusätzliche Aspekte zu berücksichtigen. Es gibt spezifische Anforderungen, die eine Low Code Plattform per se nicht abdecken kann. Hierfür bietet das A12-Team den Projekten mit einem regelmäßig aktualisierten Leitfaden eine praxisorientierte Hilfestellung.



3. Plasma UI/UX Design-System

Wiederverwendbare Widgets

Widgets sind wiederverwendbare Komponenten, die den Plasma-Design-Konventionen und UX-Konzepten folgen. Sie unterstützen Geschäftsanwendungen, die auf Desktops, Tablets und Smartphones mit Tastatur-, Maus und Touch-Eingabe laufen. Die Komponenten bieten eine einfach zu verwendende, gut dokumentierte, stark typisierte API und sind erweiterbar und anpassbar.

Features:

- ⊕ Enterprise-UI für Webanwendungen
- ⊕ Unterstützung für Desktop, Tablet und Smartphone
- ⊕ Tastatur, Maus und Touch
- ⊕ Barrierefreiheit
- ⊕ Kompatibel mit allen gängigen Browsern
- ⊕ Stark typisierte Programmier-API in TypeScript
- ⊕ Erweiterbarkeit von Designs: Von der Anwendungsebene bis hin zu jedem einzelnen Detail pro Komponente
- ⊕ Nahtlose Integration mit A12-Engines und -Services

Durchdachtes Theming-Konzept

Plasma setzt auf ein leicht erweiterbares Theming-Konzept, mit dem sich das Look & Feel einer Anwendung in Windeseile anpassen lässt. Durch die technische Umsetzung mit Styled Components und React entfällt das Laden von CSS-Dateien. Ein Theme kann ad hoc durch das Setzen einer JS-Variable aktiviert werden. Plasma enthält vier Themes out-of-the-box, die jeweils einen Dark- und Light-Mode unterstützen:

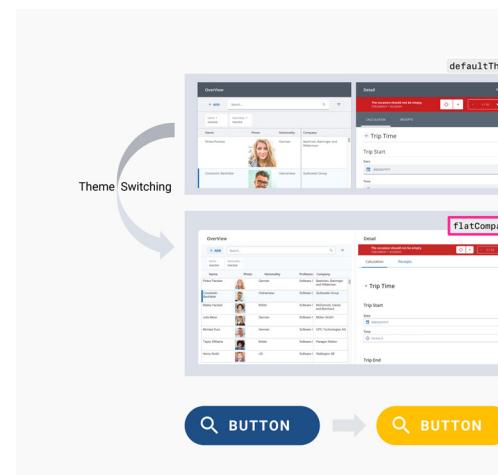
`defaultTheme` – klassisch

`flatTheme` – modern

`compactTheme` – platzsparend

`flatCompactTheme` – modern & kompakt

Themes kapseln nicht nur Stilvorgaben, sie können auch miteinander kombiniert werden. Die bestehenden Basis-Themes sind der ideale Ausgangspunkt, um projektspezifische Anforderungen abzudecken.



Thème erstellen:

Nutzung von `createTheme()` mit individuellen Konfigurationen

```
1 import { createTheme } from "@com.mgmt.a12.widgets/widgets-core"
2
3 const theme = createTheme({
4   colors: { primaryColor: "red" },
5   baseTheme: "flat-compact"
6});
```

Erweitern und anpassen:

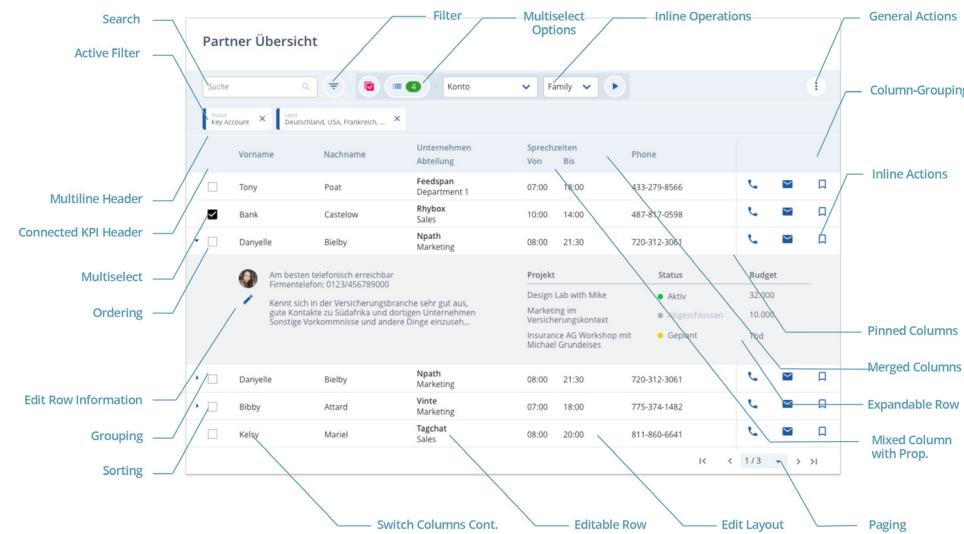
Typografie: `font`, `fontSize`, `fontWeight`

Spacing: `spacing`, `horizontalSpacing`, `verticalSpacing`

```
components: {
  button: { primary: { background: "orange" } }
}
```

Beispiel: Komponenten-Styling: gezeigt z. B. Buttons verändern

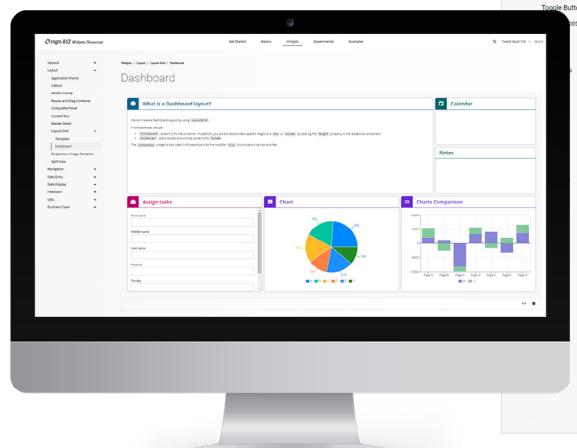
3. Plasma UI/UX Design-System



Im Gegensatz zu reinen Designsprachen wie Material Design berücksichtigt Plasma auch einen erweiterten Funktionsumfang, der typischerweise von Geschäftsanwendungen abverlangt wird. Die Abbildung zeigt beispielhaft das Gesamtkonzept für Tabellen und alle gängigen Features.

A12 Widget Showcase

<https://a12.mgm-tp.com/showcase/#/>



Der A12 Widget Showcase enthält Beispiele für alle Plasma Komponenten.

Button Widget - Widgets Showcase

https://a12.mgntp.com/showcase/#component.widgets/general/button/button/button-widget

Get Started Basics Widgets Experimental Examples

Widgets / General / Buttons / Button

Buttons

Buttons can be used to trigger different actions such as printing or exporting. As there are several use cases for buttons, A12 offers a great variety to choose from.

Primary Buttons

This example is the default primary button along with its `:active`, `:destructive`, `:inverted` and `:disabled` variants.

This example is the default primary button along with its `:active`, `:destructive`, `:inverted` and `:disabled` variants.

Secondary Buttons

This example is the default secondary button along with its `:active`, `:destructive` and `:disabled` properties applied.

Invert Buttons

The `:inverted` buttons are recommended for use against dark backgrounds.

04

Modellierungsplattform

Die A12 Modellierungsplattform stellt eine Reihe von Modellierungswerkzeugen und eine Regelsprache bereit, mit denen eine hohe fachliche Komplexität von Geschäftsanwendungen abgebildet werden kann. Die folgenden Abschnitte geben eine kurze Einführung in die Modellierungsphilosophie von A12 und stellen die wichtigsten Modelle und Werkzeuge vor.

4. Modellierungsplattform

Modellierung als neue Disziplin der Softwareentwicklung

In klassischen Entwicklungsprozessen klären Business-Analysten mit der Fachabteilung zunächst die Anforderungen an die zu entwickelnde Software ab. Anschließend beschreiben sie die Anforderungen in Prosa und geben sie an das Entwicklerteam weiter. Diese klassische Form der Anforderungsanalyse bleibt bei Projekten bestehen, die auf A12 setzen – wenn auch in einem geringeren Ausmaß. Hinzu tritt aber eine weitere Rolle: Business Analysten und Fachexperten sind durch Modellierungswerzeuge in der Lage, weite Teile der Anwendung eigenständig zu entwerfen und anzupassen. Sie erhalten einen völlig neuen Gestaltungsspielraum. **Business Analysten und Fachexperten werden zu Co-Entwicklern bzw. »Citizen Developers«.** Die nebenstehende Abbildung illustriert die Unterschiede der zwei Herangehensweisen.

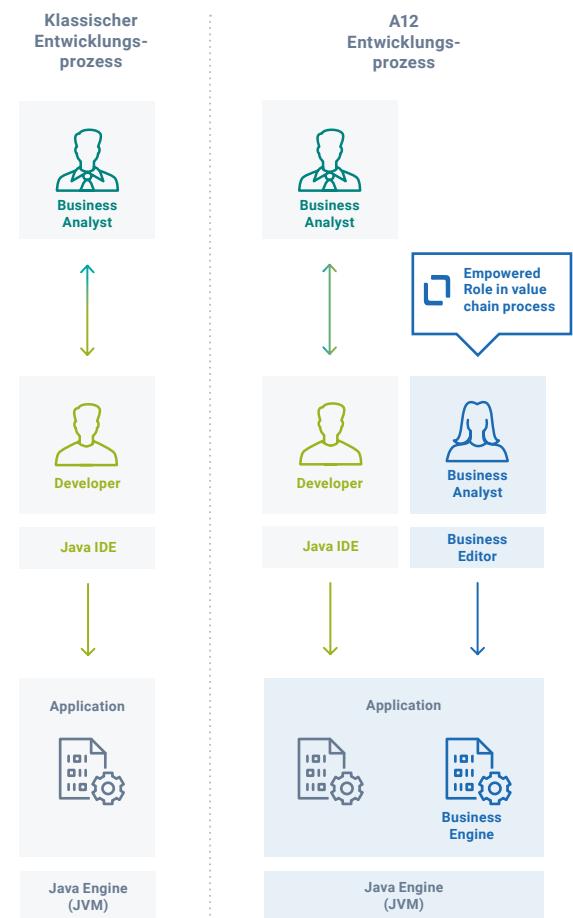
In den meisten Projekten stellt mgm die Business Analysten als Teil des Entwicklungsteams. Das hat den Vorteil, dass sie mit den Modellierungswerzeugen und -techniken bereits vertraut sind. Kundenseitige Fachexperten werden i.d.R. bereits zu Beginn eines Projekts eingebunden. Nach einer Einführung in die Modellierungswerzeuge sind sie in der Lage, wesentliche Teile der Anwendung eigenständig anzupassen.

Das Modellierungs- konzept von A12

Der Modellierungsansatz von A12 unterscheidet sich in einem wesentlichen Punkt von den Modellierungsansätzen weiterer Low Code Plattformen: A12 folgt dem Modellierungsparadigma »Data first«. Anstatt mit dem Zusammenklicken einer Benutzeroberfläche zu beginnen, startet die A12-Modellierung bei der Definition fachlicher Zusammenhänge.

Der entscheidende Vorteil dieses Vorgehens ist die klare Trennung der fachlichen Beschreibung der jeweiligen Domäne von einer bestimmten Anwendung.

So entstehen Synergien durch die Anwendungs- und kontextübergreifende Nutzung der Fachlichkeit und eine große Flexibilität für die Weiterentwicklung und Wartung langlebiger Enterprise-Software.



Links ist die klassische Rollenverteilung abgebildet, rechts die Rollenverteilung des modellbasierten Ansatzes. Der Business Analyst gestaltet neben dem Entwickler Teile der Anwendung eigenständig mit.

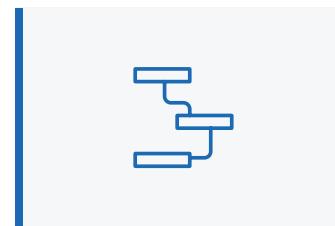
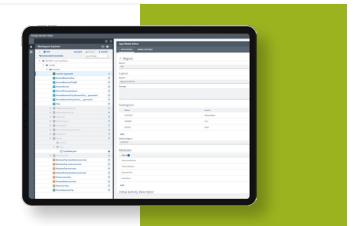
4. Modellierungsplattform

Hauptmodellierungsebenen im Überblick



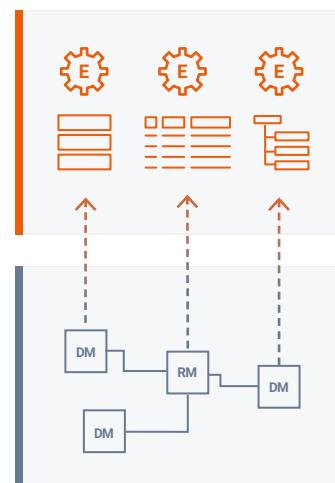
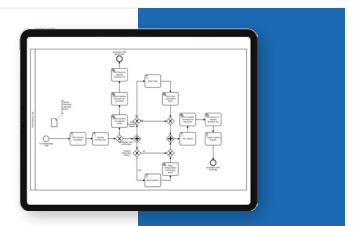
App-Modellierung

- Anwendungsrahmen



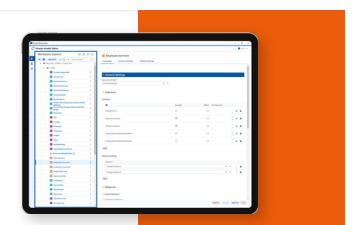
Workflow-Modellierung

- Geschäftsprozesse auf Basis von BPMN 2.0
(Business Process Model and Notation)



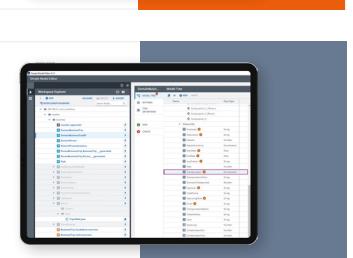
UI-Modellierung

- Form-Modelle → Formulare
- Overview-Modelle → Tabellen
- Tree-Modelle → Hierarchien
- Content-Modelle → Seiteninhalte



Daten-Modellierung

- Document Model
- Relationship Model
- Mapping Model
- Regelsprache



4. Modellierungsplattform

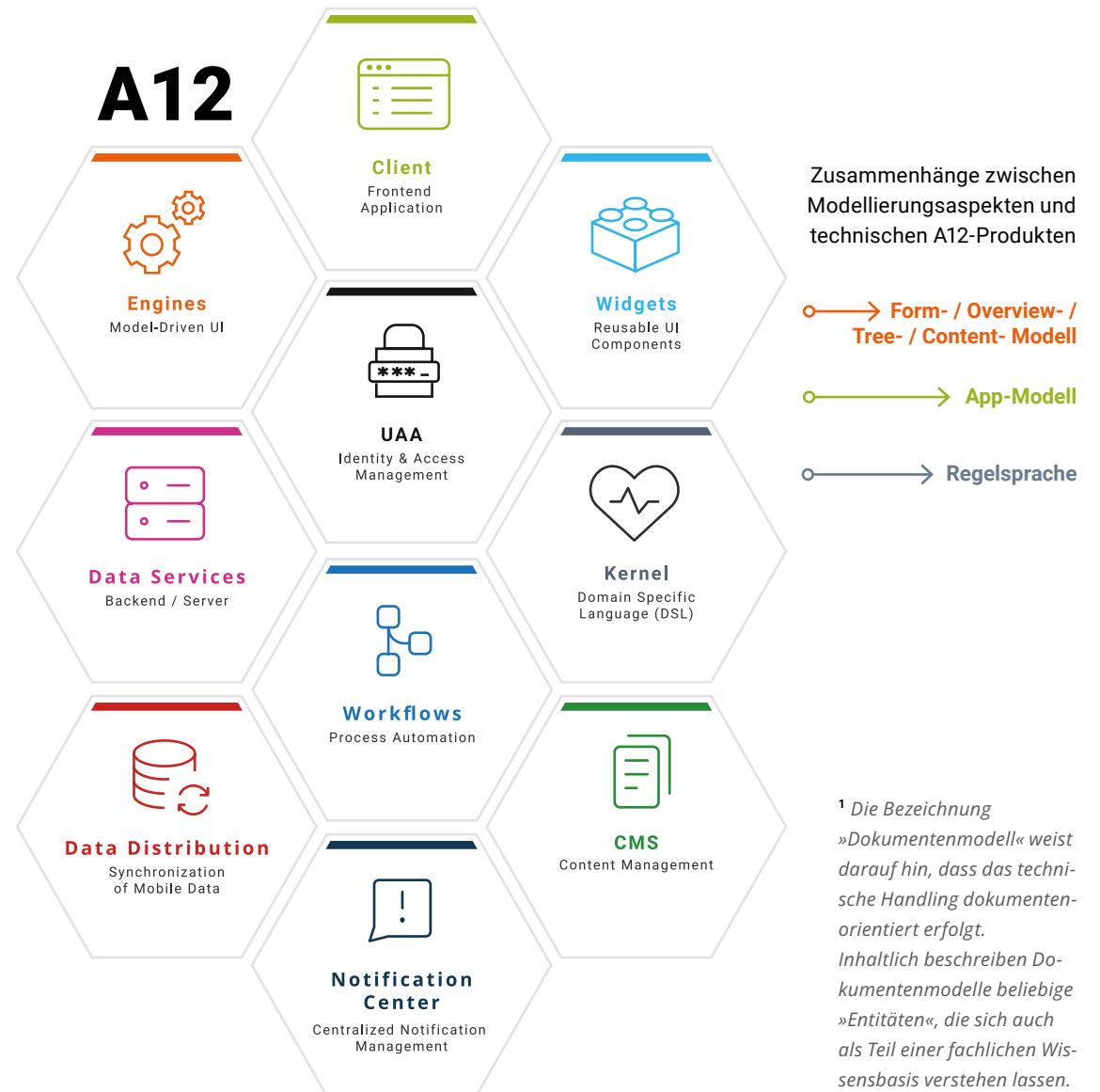
Modellierung von Fachlichkeit und Anwendungslogik

Die erste Modellierungsaufgabe besteht in der Definition von Geschäftsentitäten und ihren Beziehungen zueinander – der Datenmodellierung. Mit Hilfe von Modellierungstools können Business Analysten und Fachexpert*innen die Datenstrukturen von abgebildeten Entitäten wie Verträgen oder Produkten in sogenannten »Dokumentenmodellen«¹ erstellen und anpassen.

Darüber hinaus können sie mit einer integrierten Kernel-Sprache Validierungsregeln und Berechnungen definieren – sprich: die Anwendungslogik abbilden. Mit Relationship-Modellen können Verknüpfungen zwischen Datenmodellen beschrieben werden.

Durch die Modellierung der geschäftsspezifischen Aspekte sind Fachlichkeit und Technik voneinander getrennt. Fachliche Inhalte können angepasst werden, ohne größere technische Anpassungen vorzunehmen. Die Technik kann weiterentwickelt werden, ohne dabei alle fachlichen Inhalte anzupassen. Wir sind überzeugt davon, dass diese Trennung von Fachlichkeit und Technik die Zukunft der Enterprise Software prägen wird. Sie beschleunigt die Entwicklung, vermeidet kostspielige Neuimplementierungen und ermöglicht eine schnelle Anpassung an Veränderungen.

 Implementiert ist die Regelsprache für Validierungen und Berechnungen in der technischen A12-Komponente Kernel
→ siehe auch S. 50



4. Modellierungsplattform

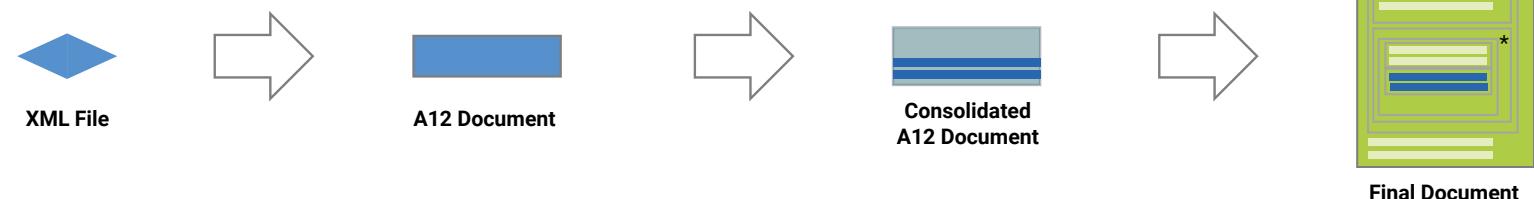
Modellierung von Daten-Transformationen und -Mappings

Überall dort, wo Daten aus verschiedenen Quellen zusammengeführt werden, sind Datentransformationen und -mappings unverzichtbar – sei es bei der Integration einer Geschäftsanwendung mit einem benachbarten System oder in großflächig ausgelegten Datenkonsolidierungsinitiativen wie der Registermodernisierung. Bislang galten diese Operationen als technische Aufgaben, die von einem Entwicklungsteam ausprogrammiert werden mussten. Mit der Transformations- und Mapping-Unterstützung von A12 werden sie jedoch modellierbar und können unmittelbar von den Fachexperten modelliert werden. Sie wissen meist ohnehin viel genauer als das Entwicklungsteam, welche Ausgangs-

daten auf welche Weise in ein Zielformat fließen und welche Endnutzerinteraktionen erfolgen sollen. Mit den Modellierungswerkzeugen, die in der integrierten Modellierungs-Umgebung von A12, dem Simple Model Editor (SME), zur Verfügung stehen, können Fachexperten die Anforderungen unmittelbar als Modelle ausdrücken. Im *Mapping-Modell* können das Ziel und verschiedene A12 Dokumentenmodelle als Quellen ausgewählt werden. Notwendige Datenkonsolidierungen können als Berechnungsregeln in der vertrauten Kernel Sprache ausgedrückt werden; die gesamte Datenbasis aus Quellen und Ziel steht dabei zur Verfügung. Im nächsten Schritt werden die konsolidierten Daten in das

Ziel-Dokument übertragen. Die Einzelmodelle dieser Schritte können unterschiedlich wiederverwendet werden. Dies erlaubt optimale Flexibilität im Umgang mit sich verändernden Quell- und Zielschemata.

Um die Daten-Pipeline zu vervollständigen, bietet A12 den *A12 Transformer*, welcher XML-Daten in A12-Dokumente und umgekehrt konvertiert. Diese erlaubt die nahtlose Integration zu Drittsystem-APIs. Das *Transformer-Modell* wird zunächst über ein Kommandozeilen-Tool in die CI/CD-pipeline integriert; perspektivisch ist aber auch die Integration in den SME geplant.



Transformer Model

- Create A12 Document Model for XML Schema
- Parse XML file into A12 Document

Precomputation Model

- Use the Kernel Language to express Computations in the joint data context

Structural Mapping

- Add data to the existing target document

Mapping Model

4. Modellierungsplattform

Modellierung von Benutzeroberflächen

Aufbauend auf den Datenmodellen sind Business Analysten in der Lage, mit den Modellierungswerkzeugen von A12 bestimmte Teile der Benutzeroberflächen zu erstellen. Die Modellierung der Oberflächen beschränkt sich derzeit auf

die Bereiche, in denen modellgetriebene Komponenten zum Einsatz kommen. Dafür stehen eine Reihe spezieller UI-Modelle bereit:



Die Modellierung folgt nicht einem *What-you-see-is-what-you-get Prinzip*. Stattdessen beschreiben die Modelle die zugrundeliegenden Strukturen der Benutzeroberfläche. Das hat den Vorteil, dass die Modelle wiederum unabhängig von der technischen Umsetzung sind. Für die tatsächliche Darstellung kommt das Plasma Designsystem zum Einsatz. Es sorgt für eine kohärente Darstellung und ein stimmiges Nutzererlebnis.

UI-Modelle beziehen sich in der Regel auf A12-Datenmodelle. Sie stellen Verbindungen zwischen den Feldern von Datenmodellen und UI-Elementen her. Nehmen wir zum Beispiel ein Eingabefeld: Ein UI-Modell beschreibt seine Position in einem Formular, seine Beschriftung und eventuell zusätzliche Benutzeranweisungen in einem Textfeld. Ein Dokumentenmodell spezifiziert den zugrunde liegenden Datentyp und die Validierungsregeln.

Für jedes UI-Modell bietet A12 eine entsprechende Engine – die Form Engine, die Overview Engine, die Tree Engine und die Content Engine. Sie bringen die Modelle in einer Anwendung zum Leben.

→ Mehr dazu auf S. 48.

4. Modellierungsplattform

Modellierung von Workflows

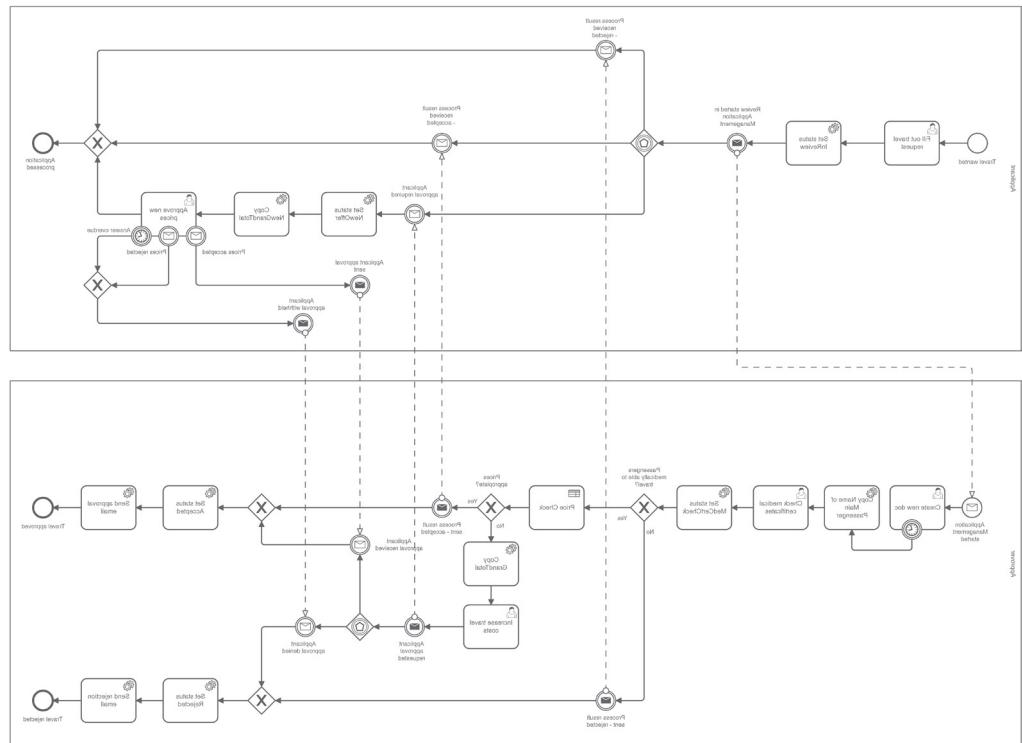
Für die Modellierung von Geschäftsprozessen setzt A12 auf BPMN 2.0 (Business Process Model and Notation), einen bestehenden, etablierten Standard. Die modellierten Geschäftsprozesse fügen sich nahtlos in das Modellierungs-Konzept von A12 ein. Dokumentenmodelle beschreiben die Daten, die von einem Prozess genutzt werden. Mit Hilfe von Formular-Modellen lassen sich die jeweiligen User-Tasks im Detail umsetzen.

Modellierung des Aufbaus einer Anwendung

Der Rahmen einer Anwendung lässt sich mit einem **App Model** definieren. Es fungiert als eine Art Container für alle weiteren Modelle.



Das App Model bietet eine Konfigurierbarkeit bestimmter Funktionalitäten der technischen Komponente **Client** (siehe S. 49).



Modellierung von Druckvorlagen

Im Rahmen von Geschäftsanwendungen stellt sich immer wieder die Herausforderung, PDF-Dokumente zu generieren – sei es ein Vertrag im Versicherungsumfeld, ein Rechnungsnachweis in einem Online-Marktplatz oder der Bescheid einer behördlichen Leistung. Mit dem in den SME integrierten Print Model Editor von A12 lassen sich Druckvorlagen erstellen, als Druck-Modelle (Print Model) abspeichern und einfach in A12-Anwendungen bringen. Die resultierenden PDFs

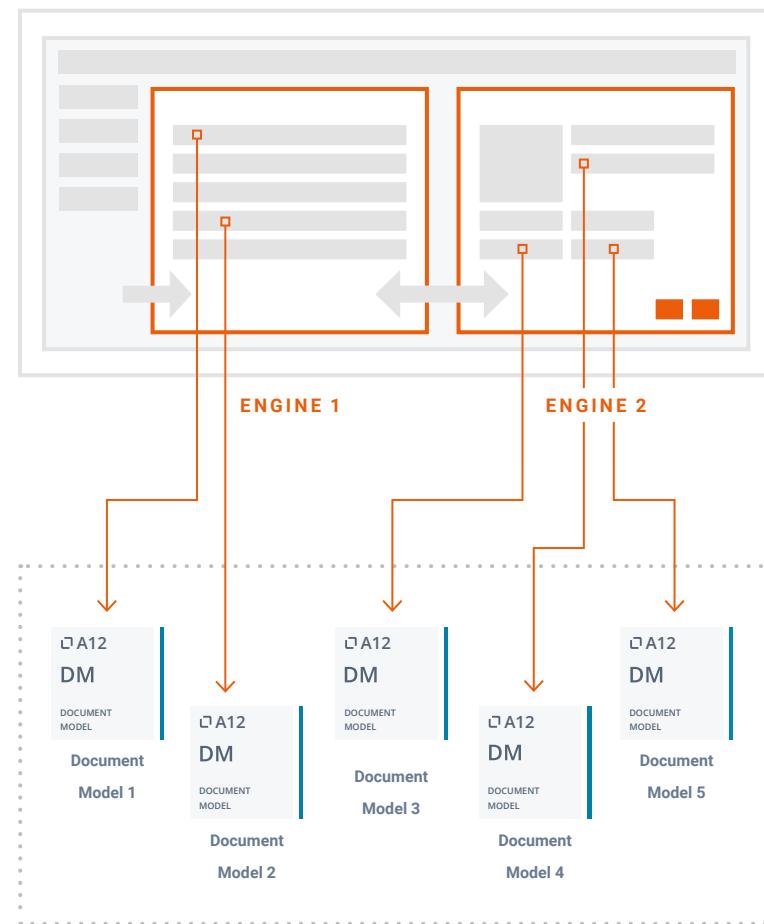
sind konform mit den Standards PDF/A-3 sowie PDF/UA und damit barrierefrei. Der Editor ermöglicht die komfortable Gestaltung von Seiten und Abschnitten, indem einzelne Elemente wie Texte und Bilder per Drag & Drop ergänzt und platziert werden. Dabei lassen sich auch direkt Felder, Berechnungen und Tabellen einfügen, die auf ausgewählte A12 Dokumentenmodelle verweisen und mit den entsprechend hinterlegten Daten gefüllt werden.

4. Modellierungsplattform

Komplexere Modellierung: Composed Document Models

Mit zusammengesetzten Dokumentenmodellen (*Composed Document Models*, kurz **CDM**) ist es möglich, **mehrere Dokumentenmodelle in einer Engine** zu nutzen – vorausgesetzt, es besteht eine Beziehung zwischen den Modellen (definiert in einem Relationship Model). Dank CDMs lässt sich ein Formular mit Daten speisen, die in ganz unterschiedlichen Modellen definiert sind. Das Modellierungskonzept von A12 gewinnt durch CDMs deutlich an Flexibilität und Ausdrucksstärke.

Der SME bietet für die Erstellung von CDMs einen eigenen Editerbereich. Auch CDM-basierte Formulare, in die Felder aus unterschiedlichen Dokumentenmodellen gebracht werden können, können direkt im SME modelliert werden. Neben modellübergreifenden Validierungs- und Berechnungsregeln erstreckt sich der CDM-Support auch über Workflows. So lassen sich zusammengesetzte Daten beispielsweise in Prozessmodellen referenzieren und Felder in Task-Listen können aus unterschiedlichen Dokumentenmodellen stammen.



4. Modellierungsplattform

Die Modelltypen von A12

KATEGORIE	BEZEICHNUNG	BESCHREIBUNG
Data Model	Document Model	A12-Dokumentenmodelle enthalten Felddefinitionen und zugehörige Validierungsregeln in einer Hierarchie aus Gruppen. Validierungsregeln reichen von einfachen Einschränkungen – z.B. der Definition von Pflichtfeldern – bis hin zu komplexen Mustern und Bedingungen über mehrere Felder hinweg.
	Relationship Model	Relationship-Modelle beschreiben Verknüpfungen zwischen Dokumenten. Sie modellieren die Beziehungseigenschaften und -beschränkungen.
Data Processing	Mapping Model	Das Mapping-Modell ermöglicht es, eine Eingangsdatenstruktur in ein Zielformat zu bringen - von der simplen Zuordnung von Feldern über Filteroperationen bis hin zu komplexen Berechnungen
UI Model	Form Model	Form-Modelle definieren die Strukturen und Inhalte von Online-Formularen. A12-Formulare bestehen aus gängigen UI-Elementen wie Eingabefeldern, Schaltflächen, Beschriftungen, Ankreuzfeldern usw. Die Modellierungswerzeuge bieten leistungsfähige Möglichkeiten, diese Elemente zu organisieren.
	Overview Model	Overview-Modelle bieten vielfältige Möglichkeiten für eine tabellarische Darstellung von Daten.
	Tree Model	Tree-Modelle erlauben es, Datenstrukturen hierarchisch darzustellen und zu bearbeiten.
Workflow	Content Model	Content-Modelle sind dafür ausgelegt, statische Inhaltsseiten abzubilden - von der Start- und Informationsseite bis hin zum Impressum
	BPMN 2.0	A12 unterstützt die Modellierung von Geschäftsprozessen im BPMN (Business Process Model and Notation) Standard. BPMN-Modelle greifen nahtlos mit A12-Modellen ineinander.
App Model	App Model	Ein App-Modell definiert den Rahmen der Anwendung und fungiert als eine Art Container für alle weiteren Modelle.
Output Model	Print Model	Das Print Model fungiert als Template für PDF-Repräsentationen eingefügter A12 Dokumente



4. Modellierungsplattform

Vorteile der »Data First«-Modellierung mit der A12-Regelsprache

Mit den A12 Werkzeugen zur Datenmodellierung können Fachexpert*innen und Business Analysten domänen-spezifische Modelle für Geschäftsanwendungen erstellen und modifizieren. Programmierkenntnisse sind nicht erforderlich. Datenmodelle kapseln zentrale Aspekte der Geschäftslogik. Sie beschreiben die Entitäten, mit denen Geschäftsanwendungen operieren – zum Beispiel Verträge oder Produkte mit all ihren relevanten Eigenschaften.

Die Nutzung von Datenmodellen bringt mehrere Vorteile mit sich:

- ⊕ Der Entwicklungsaufwand ist geringer und die Anwendungen sind anpassungsfähiger.
- ⊕ Fachexperten können Anwendungen direkt anpassen. Sie sind nicht auf das Entwicklungsteam angewiesen, das die Implementierung bei jeder einzelnen Änderung, die aus der Geschäftsdomäne kommt, überarbeitet.
- ⊕ Durch die explizite direkte Speicherung in Modellen kann nach der Fachlichkeit gesucht und recherchiert werden. Dadurch ist z.B. auch eine explizite Nachvollziehbarkeit von fachlichen Änderungen möglich.
- ⊕ Kernaspekte der Anwendung sind besser wiederverwendbar und unabhängig von Technologien.

Ein wichtiger Baustein der Datenmodellierung in A12 ist **die Regelsprache für Validierungen und Berechnungen**. Basierend auf Geschäftsanforderungen ermöglicht sie die Definition von Regeln, die alle denkbaren feldbezogenen Validierungsaufgaben abdecken. Eine möglichst umfassende Datavalidierung ist entscheidend, um Sicherheitsrisiken zu vermeiden und die Datenintegrität in Geschäftsanwendungen zu gewährleisten.

Die Regelsprache enthält viele vordefinierte Prädikate. Sie unterstützt verschachtelte Vergleiche, arithmetische Operationen und bietet spezielle Operatoren für die Behandlung bestimmter Datentypen wie zum Beispiel Datumsangaben. Darüber hinaus unterstützt sie spezielle Bedingungen, um zu überprüfen, in welchen Konstellationen Felder angegeben bzw. nicht angegeben sein dürfen. Fehlerbedingungen können sich auf mehrere verschiedene Felder und Gruppen beziehen. Die verschiedenen Teilbedingungen und Operationen sind miteinander kombinierbar.

Die Regelsprache wird direkt in den Modellierungswerkzeugen für Dokumentenmodelle unterstützt. Sie wird seit Jahren erfolgreich in großen produktiven Softwaresystemen eingesetzt. In vielen Projekten verwalteten die Kunden damit eigenständig die Validierungsregeln und Berechnungen.

Die Sprache verbindet die Einfachheit der Aussagenlogik mit der Ausdrucksstärke der Prädikatenlogik und ist besonders gut geeignet für Formulare und starke Typisierung in Fachdomänen.

Regelaingabe im SME

4. Modellierungsplattform

Die Regelsprache hat folgende Eigenschaften:

- ⊕ Regelbedingung beschreibt den Fehlerfall – dadurch können dem Endanwender präzise auf die konkrete Fehlersituation bezogene Meldungen angezeigt werden.
- ⊕ Mit Hilfe der Junktoren »Und« und »Oder« lassen sich verschiedene Teilbedingungen kombinieren.
- ⊕ Auf eine Negations-Operation wird verzichtet. Stattdessen werden die verschiedenen vordefinierten Bedingungen jeweils in positiver und in negativer Form angeboten. Dadurch wird erreicht, dass die Teilbedingungen einfacher werden und einheitlicher strukturiert zusammengesetzt werden. Die Regelbedingungen werden somit besser lesbar und klarer.
- ⊕ Quantoren der Prädikatenlogik werden nicht als formal logische Teile der Sprache angeboten, sondern implizit über Operationen. Dadurch wird erreicht, dass die Bedingungen sich an den Formulierungen des Fachpersonals orientieren und somit verständlicher werden.
- ⊕ Logische Operationen der Regelsprachen ermöglichen ein direktes Abfragen der Baum- und Wiederholungsstrukturen.
- ⊕ Mengen- und Filteroperationen über Baumstrukturen und Wiederholungsstrukturen werden unterstützt, z.B. »summiere alle Veräußerungsgewinne aus allen Aktienfonds«.
- ⊕ Iterationen über Wiederholungsstrukturen erleichtern und verkürzen die Regelbedingungen.
- ⊕ Berechnungen und Validierungen basieren auf der gleichen Sprache. Somit kann die vollständige Validierungssprache auch für Vorbedingungen der Berechnungen verwendet werden. Für die Formulierung der Berechnungsoperationen stehen alle Mengen- und Filteroperationen der Sprache zur Verfügung und es können Werte für alle vordefinierten Feldtypen berechnet werden.

Merkmale:

- ⊖ Leistungsstarke und vielseitige Validierungs- und Berechnungssprache
- ⊖ Editor mit Auto vervollständigung und Syntaxhervorhebung
- ⊖ Vordefinierte, beliebig kombinierbare Prädikate für Felder, Listen von Feldern und Gruppen
- ⊖ Arithmetische Operationen, Vergleiche, spezielle Operatoren für die Behandlung von Zeit- und Datumsangaben

MODELLIERBAR MIT A12

Fachlichkeit –
Datenmodelle mit Validierungsregeln und Berechnungen

Rahmen einer Anwendung inkl. Platzierung von modellgetriebenen Engines

Formulare, inkl. wiederholbarer Strukturen

Tabellarische Übersichten von homogenen und heterogenen Datensätzen

Baumartige Übersichten von homogenen und heterogenen Datensätzen

Beziehungen zwischen verschiedenen modellgetriebenen Komponenten

Workflows nach dem BPMN 2.0 Standard

INDIVIDUELL UMSETZBAR

komplexe Algorithmen
(z.B. generischer Prämienrechner im Versicherungsumfeld)

freie Platzierung einfacher Widgets

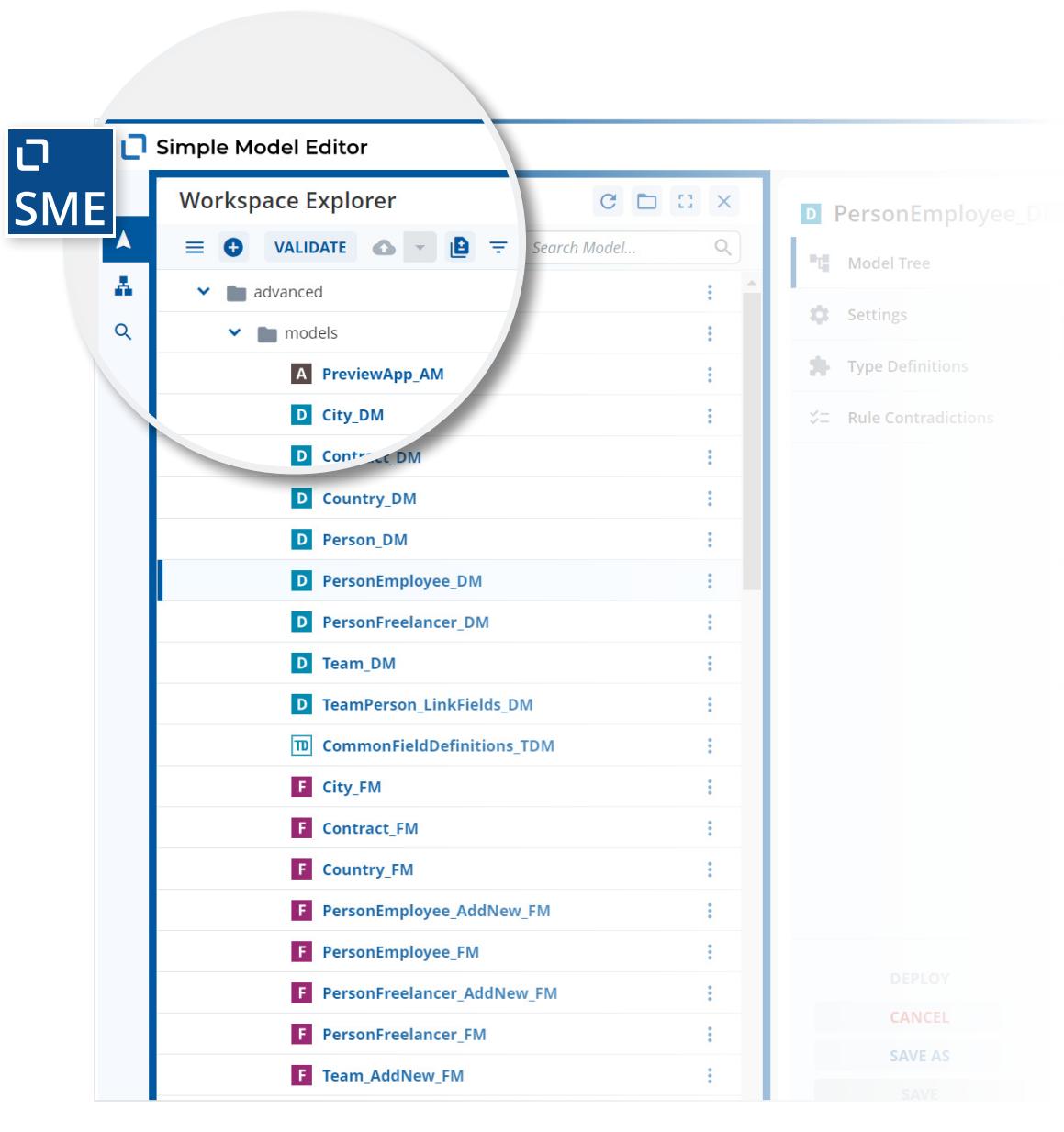
Definition oder Anpassung von Designelementen



4. Modellierungsplattform

Simple Model Editor (SME)

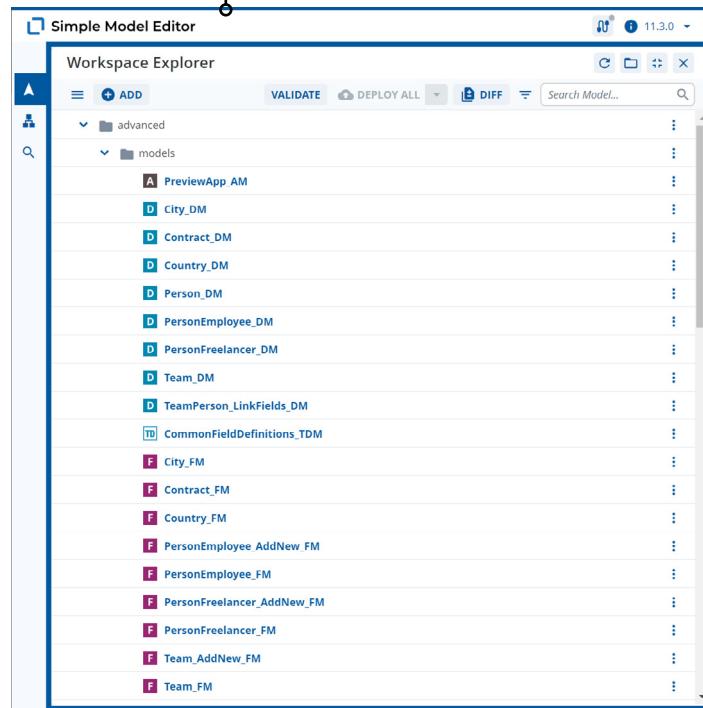
Der Simple Model Editor (SME) bildet die Schaltzentrale für die Modellierung in A12. Das Tool ermöglicht Business Analysten und Fachexpert:innen, wesentliche Teile von Geschäftsanwendungen im Handumdrehen selbst zu gestalten und zu testen – ohne Programmierkenntnisse. Eine Besonderheit: Der SME wurde als Tool für A12 selbst mit A12 gebaut.



4. Modellierungsplattform

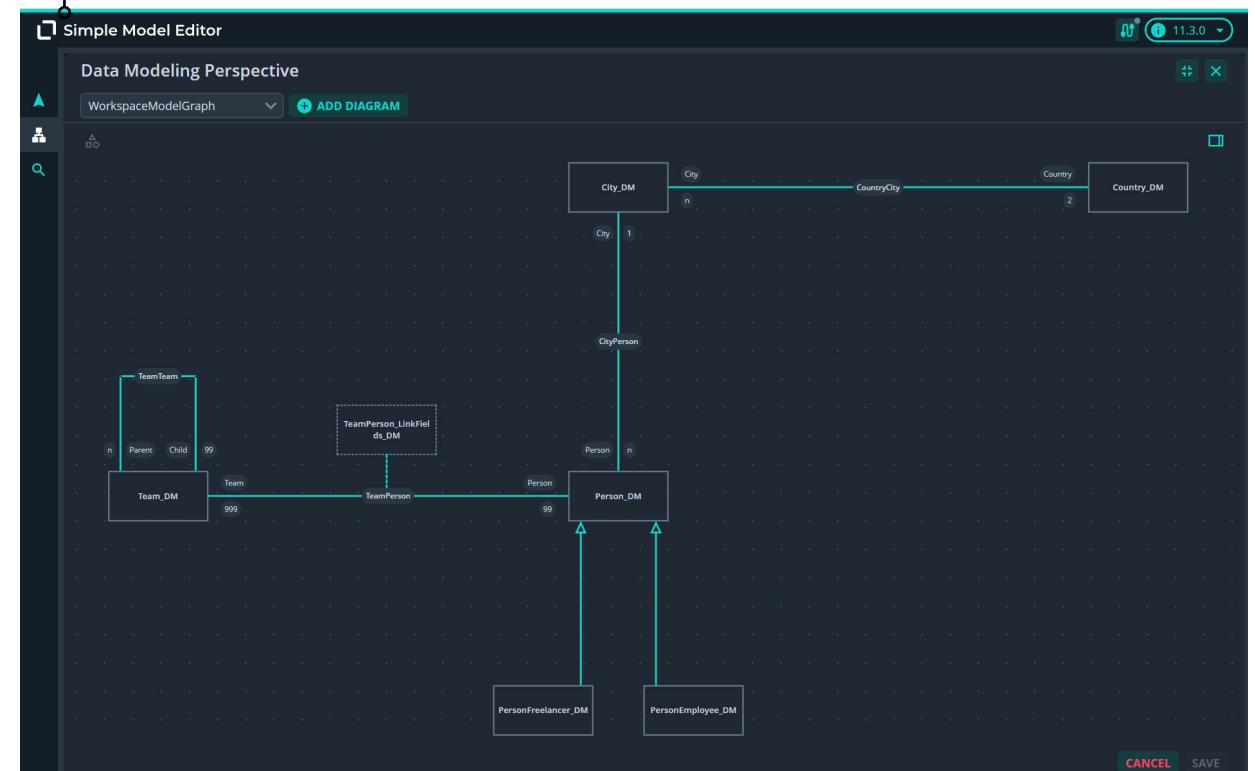
Modelle innerhalb eines Projekt-Workspace managen

- ⊕ Übersichtlicher *Workspace Explorer* für die Verwaltung aller A12-Modelle
- ⊕ Modelle erstellen, löschen, umbenennen, verschieben
- ⊕ Unterstützung für Modell-Refactorings



Fachlichkeit von Geschäftsanwendungen abbilden

- ⊕ Datenmodelle für Entitäten und Beziehungen definieren
- ⊕ Deskriptive Perspektiven: Modellbäume und Feldeditor
- ⊕ Visuelle Perspektiven: Model Graph Diagram Editor
- ⊕ Regeleditor für Validierungen und Berechnungen mit Autocomplete



Online-Formulare gestalten

- ⊕ Schnelle Erstellung von Formularen per Drag & Drop der Datenfelder
- ⊕ Live-Preview: Änderungen sind sofort sichtbar
- ⊕ Wiederholbare Strukturen für Datenabfragen mit ungewisser Anzahl

4. Modellierungsplattform

Tabellen und Baumstrukturen erstellen

- ⊕ Modellierung datengetriebener Übersichtstabellen und Baumstrukturen
- ⊕ Einfach aktivierbare Features wie Multiselection, Volltextsuche, Filter, etc.
- ⊕ Flexibel einrichtbare Schaltflächen für Custom Actions

The screenshot shows the Simple Model Editor interface. On the left, the 'Model Tree' for 'Person_DM' is displayed with fields: FirstName, LastName, Type, ContractEndDate, Photo, Name, NameComp, and WeeklyWorkhours. On the right, the 'Computation Rule Editor' shows a computation rule with a Precondition: 'AllFieldsFilled(FirstName, LastName) and A' and a Calculation: '[FirstName] + " " + [LastName]'. The calculation part includes sub-expressions like 'AllFieldsFilled' and 'AtLeastOneDateRangeOverlaps'.

Anwendungsrahmen modellieren

- ⊕ Globale Layout-Einstellungen
- ⊕ Definition von Subregionen wie Content-Bereich und Sidebar
- ⊕ Verwaltung von Modulen für wesentliche Bereiche der Anwendung

The screenshot shows the [SME] Form Model Preview. On the left, a sidebar lists options: SCREENS, VALIDATE (which is selected), RESET, WIDTH, THEME, LOCALE, READONLY, SHOW EMPTY LABELS, DATA, NOW VALUE, and CUSTOM CONDITIONS. The main area shows an 'Employee' form with sections for 'Personal Data' and 'TeamPersonTableList'. In the 'Personal Data' section, the 'Last Name*' field has a red error message: 'This field is required.' Below the sidebar, there is a navigation bar with icons for back, forward, and search.

Qualitätssicherung: Modellierte Inhalte testen

- ⊕ Ad-Hoc-Tests für das gezielte Testen ausgewählter Validierungen und Berechnungen
- ⊕ Rule Contradictions Report deckt Widersprüche in Regeln auf, die sich über mehrere Datenfelder erstrecken
- ⊕ Automatische Erkennung von Konsistenzproblemen in Modellen

4. Modellierungsplattform

Bereitstellung der Modellierungs- umgebung

Die Modellierungswerkzeuge von A12 wurden von Anfang an webbasiert entwickelt. Sie können lokal installiert sowie in der Cloud-Version als Software-as-a-Service bezogen werden.

Installer: Modellierungs- werkzeuge lokal nutzen

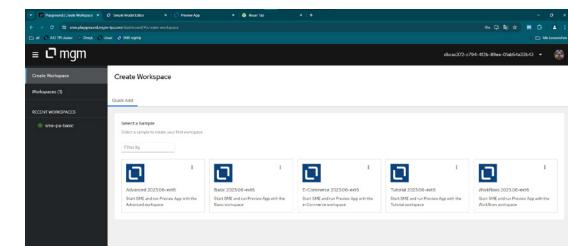
Um die Modellierungswerkzeuge von A12 einfach lokal nutzen zu können, steht der **A12 Installer** zur Verfügung. Er bündelt alle relevanten Tools in einer Installationsdatei. Der Installer wird mit jedem Release von A12 für Windows 10, macOS und Ubuntu Linux bereitgestellt.

Nach der Installation sind alle Modellierungswerkzeuge sofort einsatzbereit. Eine Reihe mitgelieferter Beispielanwendungen erleichtert den Einstieg und bildet den Startpunkt für eigene modellierte Apps. Mit Hilfe der mitgelieferten **Preview App Control** lassen sich modellierte Programme lokal im Browser ausführen.

BESTANDTEIL DES INSTALLERS	BESCHREIBUNG
Simple Model Editor (SME)	Modulares Tool, das zahlreiche Modellierungsfunktionalitäten von A12 bündelt
Camunda Modeler	Werkzeug für die Modellierung von Workflows
Preview App Control	Steuerungsapplikation für die Ausführung von A12-Anwendungen im Browser
Workspaces	Beispielanwendungen (Preview Apps), die den Modellierungsumfang demonstrieren
Dokumentation	Verweis auf bestehende Online-Dokumentation, die wahlweise auch lokal mitinstalliert werden kann.

Cloud-basierte Modellierungsumgebung

Die Modellierungsumgebung von A12 ist auch als Software-as-a-Service (SaaS) aus der Cloud verfügbar – und kann komplett ohne Installation genutzt werden. Der Zugriff erfolgt direkt über den Browser. Nachdem sich die Nutzer auf der Einstiegsseite des Cloud-Angebots eingeloggt haben, empfängt sie eine Administrationsoberfläche – die *Cloud Modeling Control*. Hier lassen sich neue *Cloud Modeling Environment Workspaces* (CME-Workspaces) anlegen und Bestehende starten. Für neue CME-Workspaces stehen die bekannten Sample Workspaces aus der lokalen Modellie-



lungsumgebung als Basis bereit. Nach dem Starten eines Workspaces öffnet sich der Simple Model Editor (SME) in einem neuen Tab und die Preview Application ist erreichbar. Der SME unterstützt offiziell die Browser Chrome (Windows, MacOS), Edge (Windows) und Chromium (Ubuntu).

05

Laufzeitplattform

Die A12 Laufzeitplattform besteht aus einer Reihe von modularen Client- und Serverseitigen Komponenten in einer modernen Enterprise Architektur. Sie stellt robuste Komponenten für typische Anforderungen von Enterprise Anwendungen bereit. Gleichzeitig lässt sie dem Entwicklungsteam durch feingranulare Einstiegs-punkte aber die volle Kontrolle, um eigenen Code einzuklinken und individuelle Anforderungen des Projekts umzusetzen.

5. Laufzeitplattform

Verändertes Aufgabenspektrum für das Entwicklungsteam

Im Vergleich zu konventionellen Individualsoftwareprojekten bringt der modellgetriebene Ansatz eine Reihe von Veränderungen für das Entwicklungsteam mit sich. Es ist nicht mehr alleine für den Bau der gesamten Anwendung verantwortlich. Der Arbeitsanteil wird vor allem in Hinblick auf den Umgang mit fachlichen Änderungen minimiert. Wenn man die Anwendung mit einem Bühnenstück vergleicht, dann sind die von den Business Analysten designeten Modelle die Protagonisten, die im Rampenlicht stehen. Die Entwickler jedoch ermöglichen es erst, dass das Stück überhaupt aufgeführt werden kann. Sie bereiten die Bühne auf und sorgen dafür, dass die Protagonisten im besten Licht stehen.

Entlastung durch modellierte Fachlichkeit

In einem konventionellen Softwareprojekt ist das Entwicklungsteam alleine für das Schreiben des gesamten Codes verantwortlich. Das setzt voraus, dass das Team die Idee der Anwendung bis ins kleinste Detail verstanden hat. Gerade bei komplexen Anwendungsbereichen wie der Steuer oder der Industriever sicherung ist das eine gewaltige Herausforderung.

Durch den modellgetriebenen Ansatz ändert sich diese Situation.

Business Analysten und Fachexperten bilden die Fachlogik in Modellen ab und bringen sie direkt in die Software.

Für Entwickler ist das eine gewaltige Entlastung. Sie müssen die modellierten fachlichen Aspekte nicht mehr verstehen und händisch implementieren. Die Arbeitsschwerpunkte verschieben sich.

Applikationsplattform anbinden, warten und erweitern

Projekte, die auf A12 setzen, starten nicht auf der grünen Wiese. Sie bauen auf einem bestehenden Fundament auf. Und dieses Fundament ist nicht statisch, es entwickelt sich stetig weiter. Eine zentrale Aufgabe der Entwickler besteht darin, das Fundament in Form der A12 Applikationsplattform für das Projekt anzubinden, zu warten und bei Bedarf individuell zu erweitern. Unterstützung bietet dabei das Technical Professional Services Team.

Komplexere Funktionen und Integrationsarbeiten

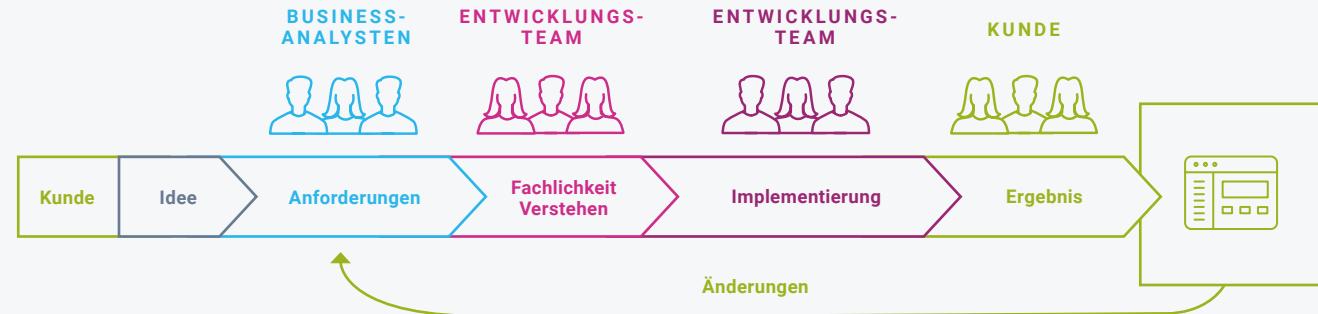
Ein weiterer Aufgabenbereich der Entwickler ist das Schreiben von Code, der komplexere Funktionen umsetzt. Das kann beispielsweise eine komplexe Berechnung sein, die den bestehenden Umfang der Modellierungswerkzeuge sprengt. Darüber hinaus ist eine der verbleibenden Hauptarbeiten die Integration in die bestehende heterogene IT-Landschaft.



5. Laufzeitplattform

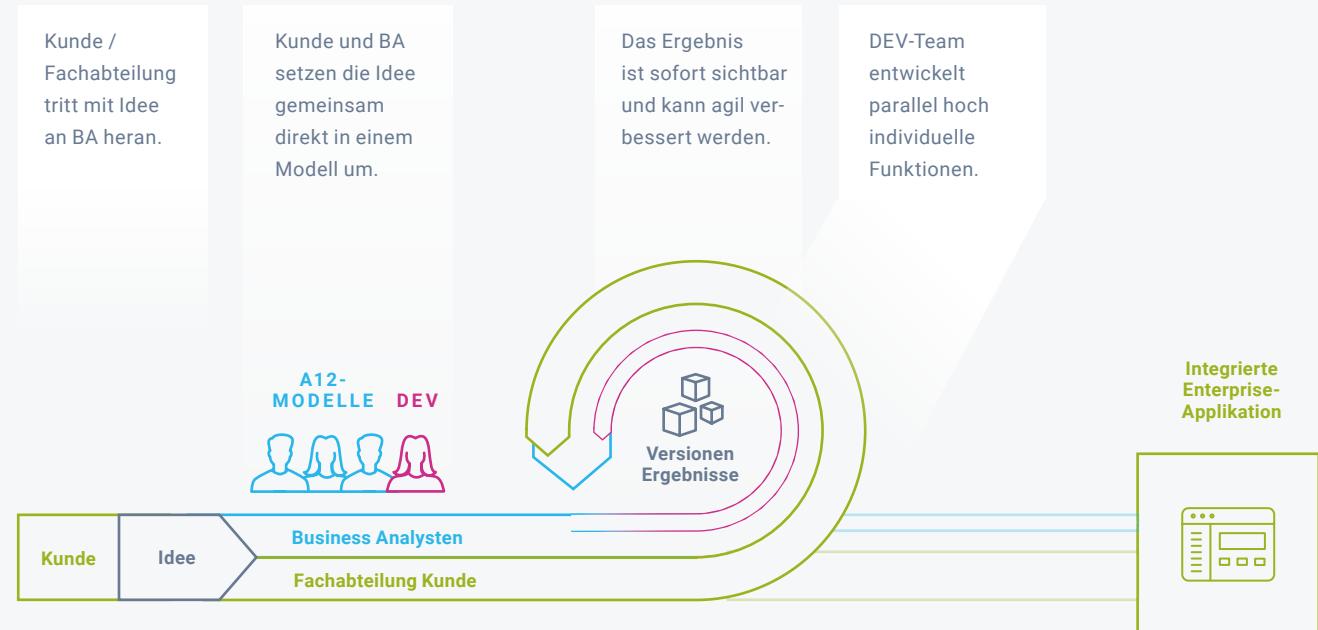
Klassischer Entwicklungsansatz vs. Modellbasierte Entwicklung

Klassischer Ansatz



Modellbasierter Ansatz

- Anforderungen in Modellen
- Gemeinsam entwickeln
- Ergebnis sofort sichtbar
- Agil und schnell



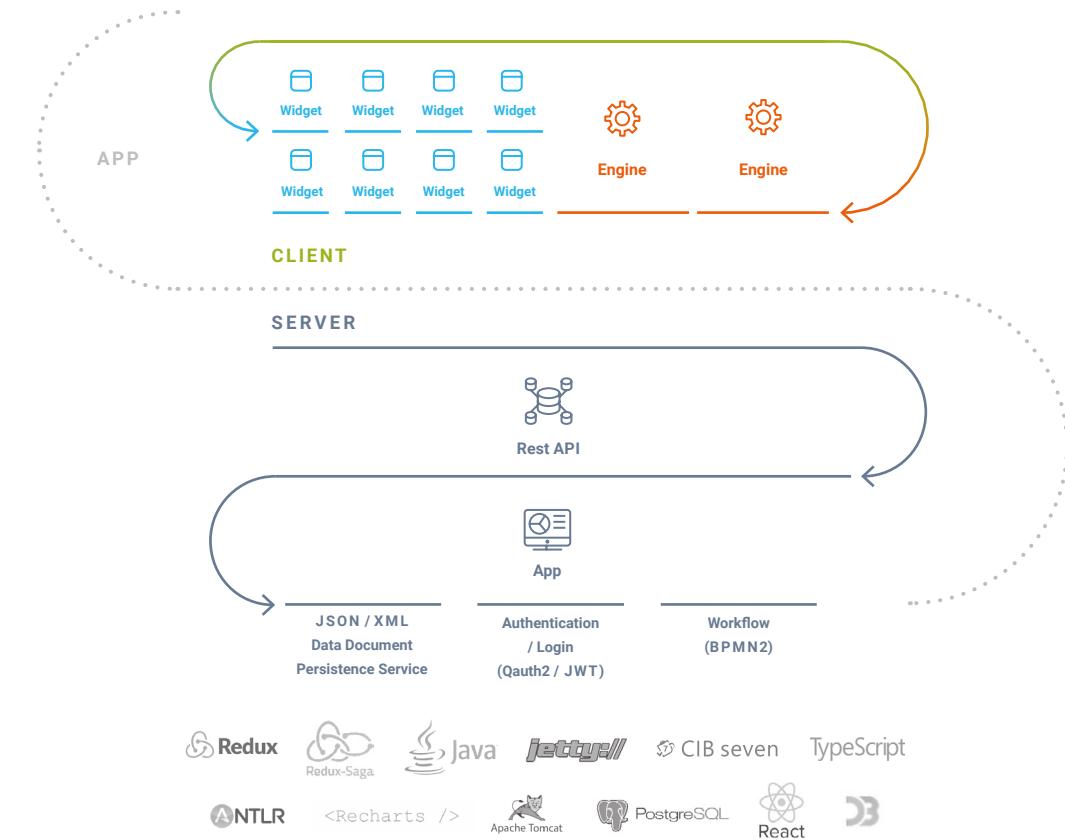
5. Laufzeitplattform

Architektur

Der Entwicklungsprozess von Geschäftsanwendungen ist kontinuierlich von technologischen Veränderungen geprägt. A12 stellt sich diesen Herausforderungen und bietet eine Laufzeit-Plattform für moderne, webbasierte Geschäftsanwendungen. Ausgehend von einem robusten Kern und modularen Lösungsbausteinen entwickeln wir diese Plattform auf allen Ebenen ständig weiter. Dafür adaptieren wir neue Technologien und Paradigmen, sofern sie auf das Ziel einzuholen, die Entwicklung hochqualitativer Geschäftsanwendungen einfacher, effizienter und nachhaltiger zu gestalten.

Eine wichtige Eigenschaft des Low-Code-Ansatzes kommt uns dabei zugute: viele der komplexesten und wichtigsten Aspekte der Anwendung werden in A12 modelliert und lassen sich damit weitgehend technologie-neutral ausdrücken. Tatsächlich überleben auch komplexe Formular-Lösungen den Technologie-Wechsel von JSPs und XForms (2012 und früher) über Angular (circa 2015) zu React (ab 2017). Der nötige Unterbau – die UI-Engines als Laufzeit-Interpreter von Modellen – wechselt, die Modelle bleiben bestehen.

Das **A12 Client-Framework** adressiert die Komplexität und die Herausforderungen moderner Webanwendungen nach dem Single-Page-Application (SPA) Ansatz. Es ist auch die Grundlage für den schnellen Aufbau modularisierter Frontends (Microfrontends). Es nutzt den modernen und bewährten React/Redux-Technologiestack, integriert A12-UI-Komponenten wie Engines und Widgets und interagiert mit den A12 Backend-Diensten wie A12-Data Services und Workflow mittels REST-APIs. Daten und Modelle sind JSON-Datendokumente. Die Anbindung kundenspezifischer Backends ist leicht möglich, wie auch insgesamt die meisten Aspekte des A12 Client-Frameworks durch Extension-Points ange-



passt oder auch überschrieben werden können. Die **serverseitigen A12-Dienste** bieten u.a. die Data-Services für die Aspekte Datenhaltung, Suche, und Modell-Repository, als auch Workflows (CIB Seven/BPMN 2), Authentifizierung/Anmeldung (LDAP, SAML, OpenID Connect, OAuth 2, JWT) und Benutzer-/Rollenmanagement. Die Dienste bauen auf

Spring Boot auf und können out-of-the-box verwendet, aber auch leicht um kunden-spezifischen Code erweitert werden. Dahinter stehen unterstützende Open Source-Produkte, darunter Postgres als Datenbank, Keycloak für das Zugriffsmanagement und Single Sign-On (SSO) sowie CIB Seven als Workflow-Engine.

5. Laufzeitplattform

Dokumentenorientierter Datenzugriff und Model-Graph

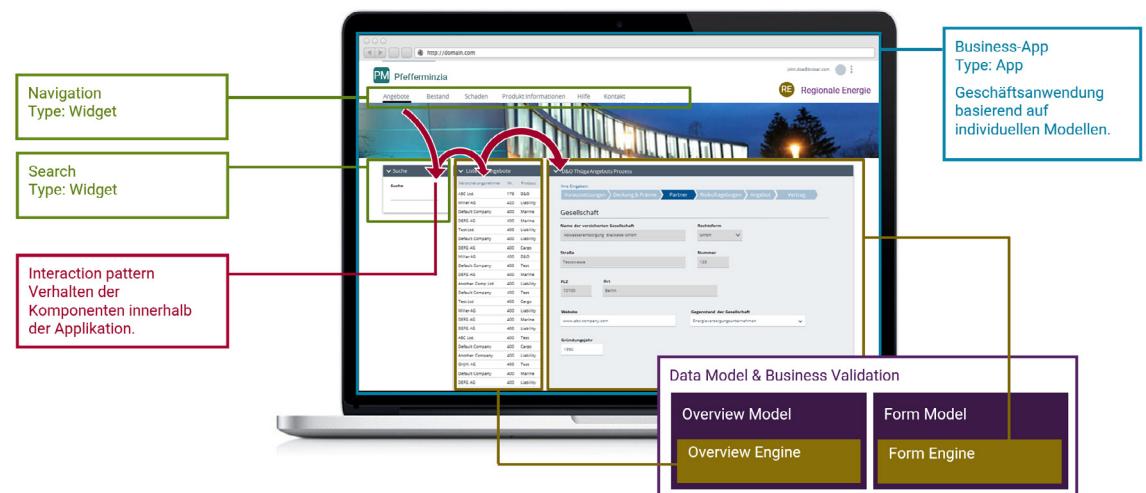
Die A12-Architektur basiert auf dem Konzept von hierarchischen Sammlungen von Feldwerten in JSON-Dokumenten (*kurz: Dokumente*). Auf diese Dokumente können Clients zugreifen und sie speichern. Dokumentenmodelle (**Schemata**) spezifizieren nicht nur Feldtypen, sondern auch Validierungs-/Integritätsregeln und Berechnungen in unserer sehr ausdrucksstarken Kernel DSL (**Domain-Specific Language**). Diese Regeln werden etwa im Rahmen der Formular-Bearbeitung automatisch von der Form-Engine ausgewertet. Mit der Query API der A12-Komponente Data Services lassen sich Anfragen auf A12-Dokumente, Links und Datagraphen sehr präzise und datensparsam formulieren. Die Anfrageergebnisse können sortiert, paginiert und auf bestimmte Felder beschränkt werden.

Daneben gibt es folgende Erweiterungen:

- ⊕ Beziehungen (Relationships) zwischen Dokumenten werden voll unterstützt; Dokumente können verknüpft und Beziehungseigenschaften und -beschränkungen modelliert werden und werden von den A12 Data Services durchgesetzt. Weiterhin gibt es ein Vererbungskonzept (Subtyping) für Dokumentmodelle. Dadurch lassen sich komplexere Domänen als Graph von Dokumentenmodellen ausdrücken; wir nennen dies den Model-Graph. Unsere Tree-Engine nutzt den Model-Graph etwa zur Darstellung von verlinkten Dokumenten in einer Baumansicht.
- ⊕ Dank der erwähnten **CDMs** lassen sich Sichten (Views) auf den Model-Graph abfragen, analog zu GraphQL.

- ⊕ **Batches:** Die A12 Data Services API bietet einen Batch REST-Endpoint zur transaktionalen Bündelung von mehreren Dokumentoperationen, etwa Anlegen eines neuen Dokuments mit gleichzeitiger Verlinkung zu einem anderen Dokument. Es gibt auch eine Operation zur partiellen Modifikation von Dokumenten, um den Netzverkehr zu reduzieren.

auf React und nutzt Redux für das State Management und Caching. Das Framework bietet eine Vielzahl von **Integrationen**: eine Datenzugriffs-Abstraktion „Data Provider“ mit Voreinstellung für A12-Data Services, die Anbindung von Process-Engines mit Voreinstellung für CIB Seven/A12 Workflow (etwa Task-Listen), einen A12 Data Distribution Client (Data-Sync, Offline-Fähigkeit), Benachrichtigungen über das



A12 Frontends

Die A12-Architektur legt einen starken Schwerpunkt auf die Vereinfachung der clientseitigen Anwendungsentwicklung. Sie bietet einen in der Praxis bewährten Anwendungsrahmen, der vom A12 Client-Framework bereitgestellt wird, sowie Engines für die Arbeit mit Modellen und Widgets für wiederverwendbare UI-Komponenten. Der Anwendungsrahmen nutzt ein Anwendungsmodell zur Steuerung des Zusammenspiels der Engines, etwa in einem Master/Detail-Kontext. Das in TypeScript geschriebene A12 Client-Framework basiert

Notification Center. Zudem bietet das A12 Client Framework viele nützliche und leistungsfähige **Features** wie asynchrone Ablaufsteuerung mittels Redux Saga, Dirty Handling und Undo Mechanismen, URL-Routing, eine Layout-Provider Abstraktion mit responsive Defaults für Desktop- und mobile Geräte, Lokalisierung. Ein A12 Frontend Client kann gemäß dem Microfrontend-Muster modularisiert werden. Dazu nutzen wir technisch „Module Federation“ von Webpack und haben darauf basierend eine Anwendungsmodul-Registry entwickelt, wodurch die dynamische Integration dieser Module möglich wird, etwa gemäß der Benutzerrollen.

5. Laufzeitplattform

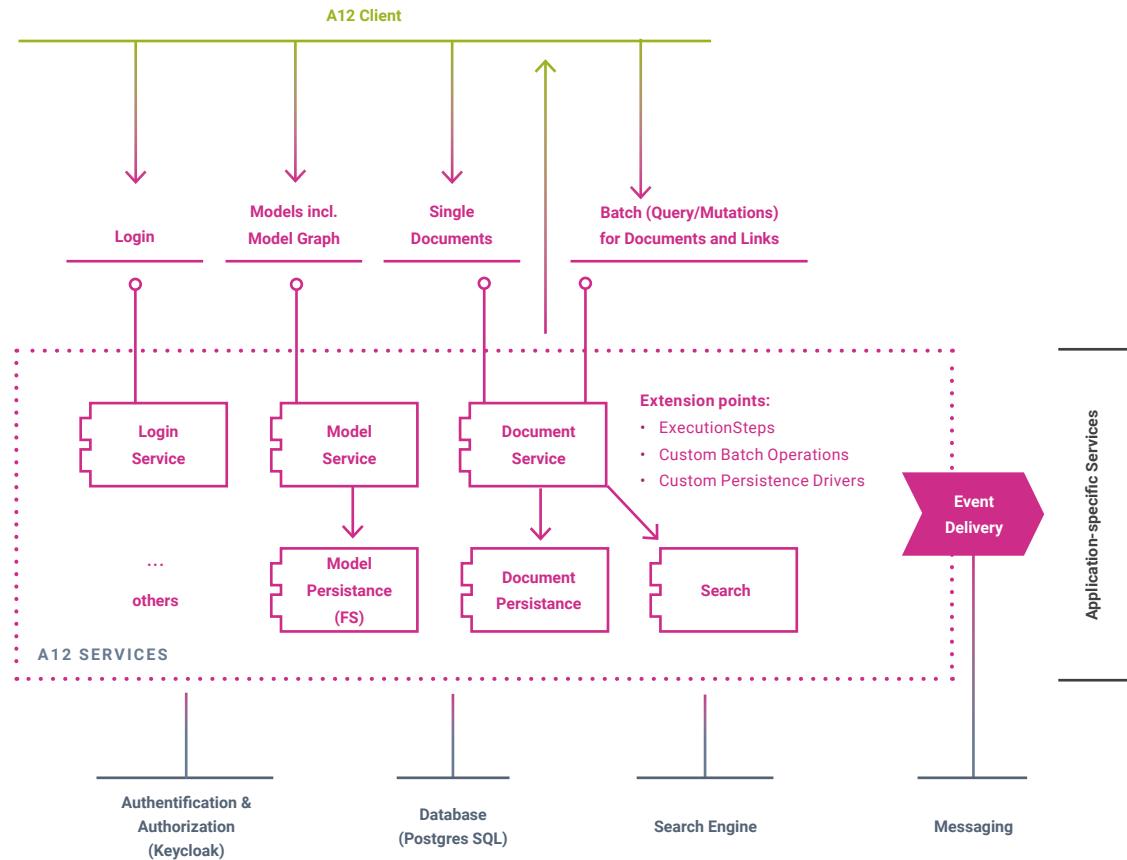
A12 Backend Services

Der zentrale Dienst ist A12 Data Services. Er bietet Zugriff auf Modelle und Dokumente und übernimmt auch die Anmeldung/Login mit SSO-Unterstützung und optionaler Keycloak-Integration (LDAP, SAML, OpenId Connect, OAuth 2, JWT). Die APIs sind als **zustandlose REST-Endpoints** und auch in Java verfügbar. Die Persistenz der Datendokumente wird durch eine Reihe zuverlässiger Technologien wie PostgreSQL unterstützt.

A12 Data Services nutzt wie alle anderen serverseitigen A12-Dienste das **Spring Boot-Framework** und steht in drei Formen bereit: als Standalone Anwendung, als Spring Boot Projekt für projektspezifische Anwendung mit kundenspezifischem Code, oder auch als Library, um in bestehenden Spring-Anwendungen punktuelle Features zu nutzen. Dazu gibt es zahlreichende **Erweiterungspunkte** und ein umfassendes Event System zur einfachen Integration von eigenem Handling-Code vor und nach Operationen.

Zur Skalierung können die Services in einem Hazelcast Cluster betrieben werden. Ein solcher Cluster kann dann unter Kubernetes dynamisch an die Last angepasst werden. Unser A12 Project Template bietet hierzu bereits Konfigurationen.

Weitere serverseitige A12-Dienste sind u.a. A12 Workflow (basierend auf CIB Seven/BPMN 2) und A12 User Management Service mit IDP-Unterstützung (Keycloak). Zudem gibt es mit A12 Data Distribution eine hochskalierbare Datenverteilungs- und Sync-Lösung mit Offline-Fähigkeit der Clients. Der Notification Service nutzt A12 Data Distribution zur Zustellung der Benachrichtigung. Der A12-Kernel kommt auf der Client- und der Server-Seite zum Einsatz.



Er validiert Daten und berechnet abgeleitete Daten auf der Grundlage von Regeln und Feldtypen, die in Datendokumentenschemata (sog. Dokumentenmodelle) beschrieben sind. Die Codegenerierung gewährleistet nativen Code für Client und Server. Im Front-End werden die Regeln und Be-

rechnungen als natives JavaScript ausgeführt und ermöglichen eine sofortige Rückmeldung an den Benutzer während der Formularbearbeitung.

5. Laufzeitplattform

Projektszenario für den Einsatz von A12

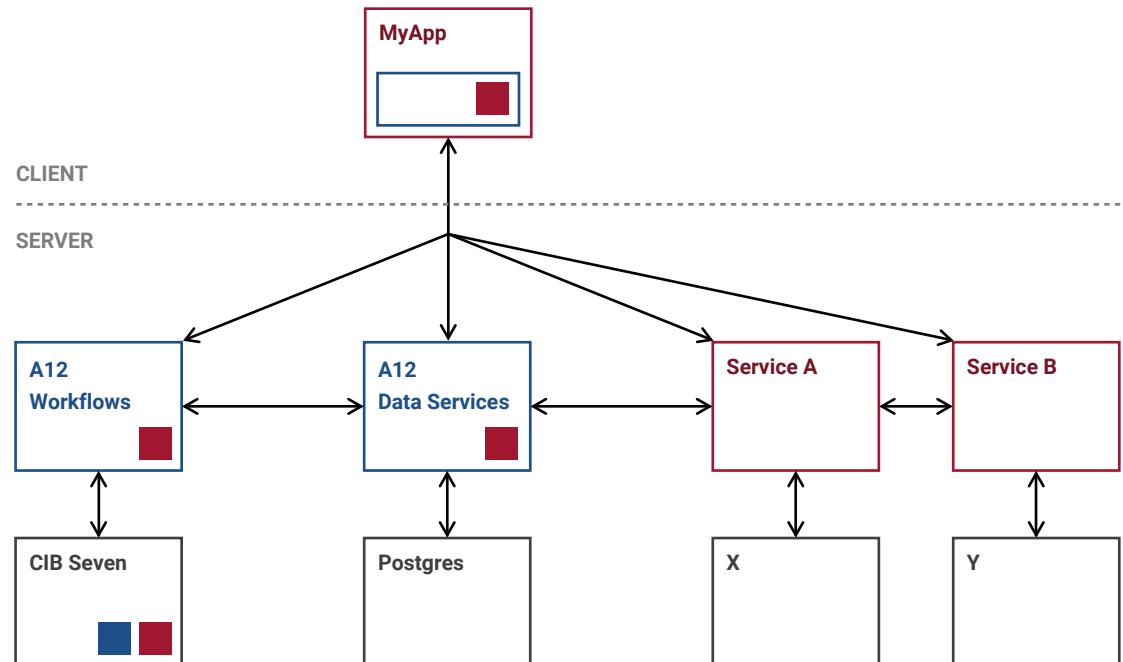
Dank des modularen Aufbaus lässt sich A12 sehr flexibel einsetzen und ist auch für Microservice Architekturen und Microfrontends dank umfangreicher Unterstützung bestens geeignet. Das folgende Schaubild demonstriert beispielhaft, wie die Bausteine von A12 mit projektspezifischen Erweiterungen und Services sowie Drittkomponenten in einem Microservice Kontext zusammenspielen können:

Zum Frontend: Die entstehende Webanwendung setzt sich dynamisch aus mehreren Teilen und entsprechenden Frontend-Projekten zusammen: der Application Shell und zwei Microfrontends, die von den projekteigenen Microservices (A und B) bereitgestellt werden.

Dabei kommen A12-Komponenten zum Einsatz: die Widgets und Engines werden individuell angepasst und das A12 Client Framework um die jeweiligen Projektanforderungen entsprechend erweitert. So kann man etwa Daten von den REST-APIs der eigenen Microservices abfragen und per Data Provider Abstraktion als JSON-Dokumente aufbereiten und so für die Engines zugänglich machen.

Auf der **Server-Seite** interagieren

- A12 Services mit optionalen projektspezifischen Anpassungen und
- Beliebige projektspezifische Services (etwa als Microservice) mit oder ohne A12-spezifischen Erweiterungen (etwa der A12 Kernel als Library oder A12 Data Services als Dependency).



- Projektspezifische Services
- A12 Services
- 3rd-Party Services

5. Laufzeitplattform

Komponenten

Die Laufzeitplattform von A12 ist modular aufgebaut und besteht aus einer Reihe lose gekoppelter Komponenten. Sie lassen sich je nach Situation flexibel – auch einzeln – im Projekt einsetzen.

Die meisten Projekte nutzen den Dreiklang aus Client, Engines und Widgets. Einige Projekte verwenden die im Data Services-Modul bereitgestellten Backend- und Server-Dienste. Andere schreiben je nach Anforderungen einen eigenen Server.



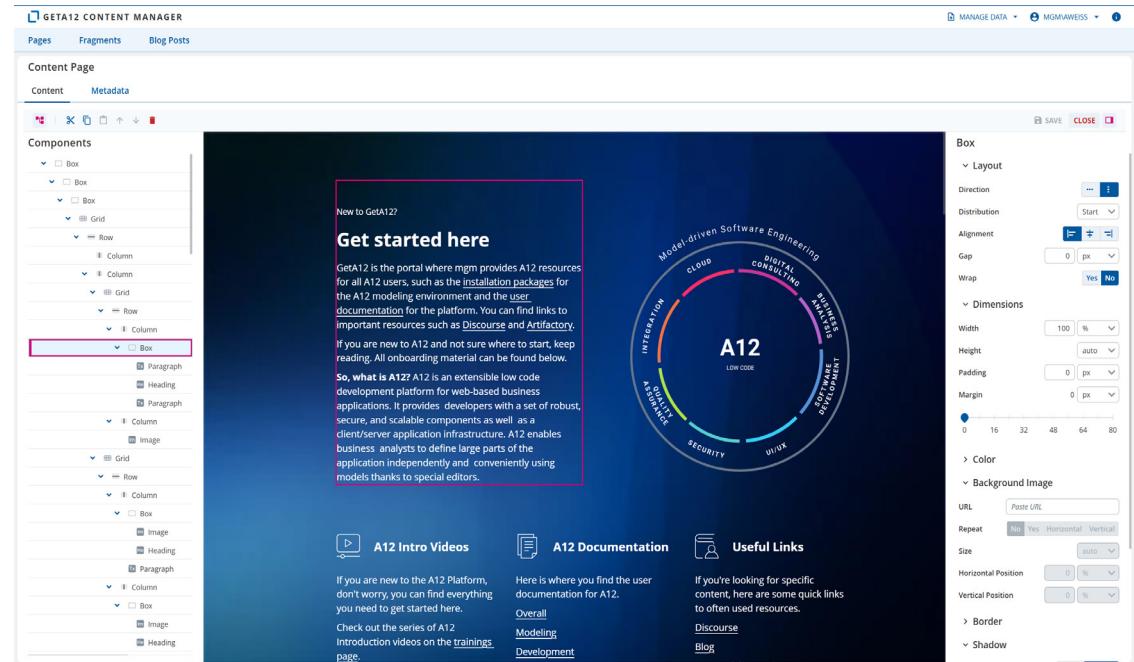
5. Laufzeitplattform

KOMPONENTE	KURZFORM	BESCHREIBUNG
Client	C	Modellgetriebene, Clientseitige Laufzeit-Komponente. Setzt das UI/UX-Konzept des Plasma Design Systems um und unterstützt Desktop, Tablet und Smartphone. Hauptaufgaben sind die Orchestrierung anderer UI Komponenten, insb. der A12 Engines, Datenbeschaffung und Zustandsverwaltung.
Engines	E	Modellgetriebene UI-Komponenten. Engines interpretieren Daten- und UI-Modelle. Sie basieren auf den Plasma UI/UX-Konzepten und nutzen die Widgets für das Rendering.
Widgets	W	Widget Library, basierend auf Plasma UI/UX-Konzepten. Siehe auch → A12 Widget Showcase .
Kernel	K	Bündelt alles für die Erstellung und Verarbeitung von Dokumenten-modellen: Modellierungswerzeuge, Sprache für Validierungen und Berechnungen, Client- und Serverseitige Laufzeitkomponenten, Java- und Typescript-API.
Data Services	DS	API für die Verwaltung von Modellen und Daten. Enthält außerdem Routinen für Client/Server Kommunikation, Validierung, Persistenz, Indizierung.
User Management, Authentication and Authorization	UAA	Bündelt Lösungen rund um Authentifikation (Keycloak, OAuth 2.0, SAML, LDAP), Autorisierung (Spring Security, RBAC, ABAC, custom Logik) und User Management.
Workflows	WF	Integration von Business Process Model and Notation (BPMN) in A12; ermöglicht grafische Modellierung serverseitiger Workflows und ihre Ausführung.
Data Distribution	DD	Transportschicht für die Synchronisation von Daten
Notification Center	NC	Mitteilungszentrale für Nachrichten wie Aufgaben, Terminen und Erinnerungen
Content Management System	CMS	Leichtgewichtiges Content Management System

5. Laufzeitplattform

CMS Content Management System (CMS)

Moderne Webanwendungen bestehen immer auch aus textuellen und bildlichen Inhalten – sei es in Form des Impressums bei öffentlich zugänglichen Webangeboten, einer individuell gestalteten Startseite oder einer News- und Blog-Sektion. Um solche Inhalte zu verwalten, bietet A12 ein neues, leichtgewichtiges Content Management System (CMS) ein. Es stellt eine schlanke und integrierte Lösung für A12-Anwendungen bereit, die redaktionelle Inhalte enthalten. Das CMS besteht aus einem Editor, mit dem sich Seiten verwalten und gestalten lassen sowie einer Content Engine als Laufzeitkomponente, welche Content-Modelle interpretiert.

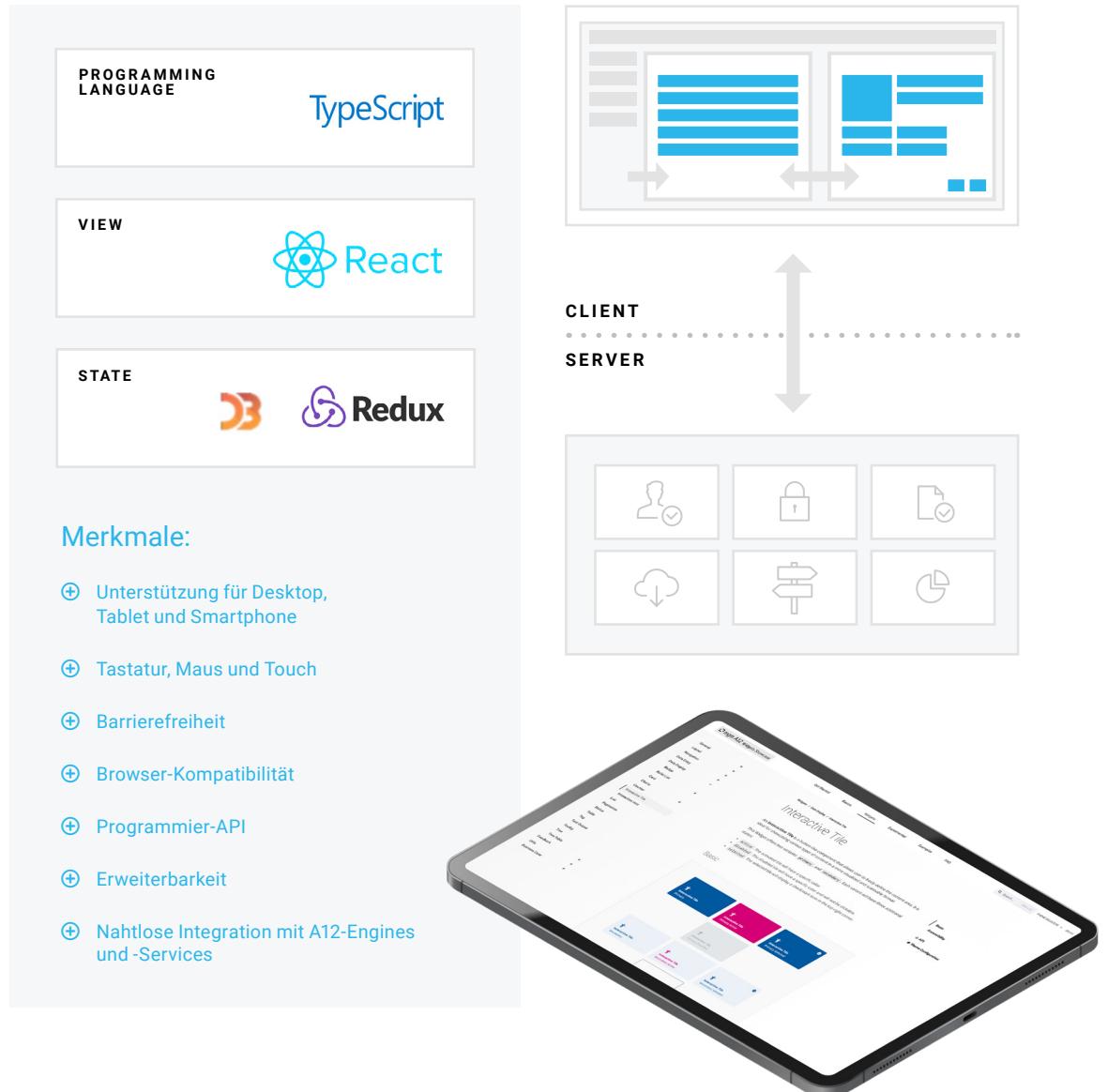


The screenshot shows the GETA12 Content Manager interface. On the left, there's a sidebar titled 'Components' with a tree view of page structures, including 'Box', 'Grid', 'Row', 'Column', and 'Image'. The main area displays a dark-themed preview of a page. The page features a central 'Get started here' section with text about the A12 portal and its resources. To the right of this is a circular graphic with the text 'A12 LOW CODE' in the center, surrounded by various business and technology terms like 'Model-driven Software Engineering', 'CLOUD', 'DIGITAL CONSULTING', 'BUSINESS ANALYTICS', 'SOFTWARE DEVELOPMENT', 'UI/UX', 'SECURITY', 'INTEGRATION', 'SUBSTANCE', and 'ASSESSMENT'. On the far right, there's a detailed configuration panel for a 'Box' component, covering settings for Layout, Dimensions, Color, and Background Image.

5. Laufzeitplattform

W Widgets

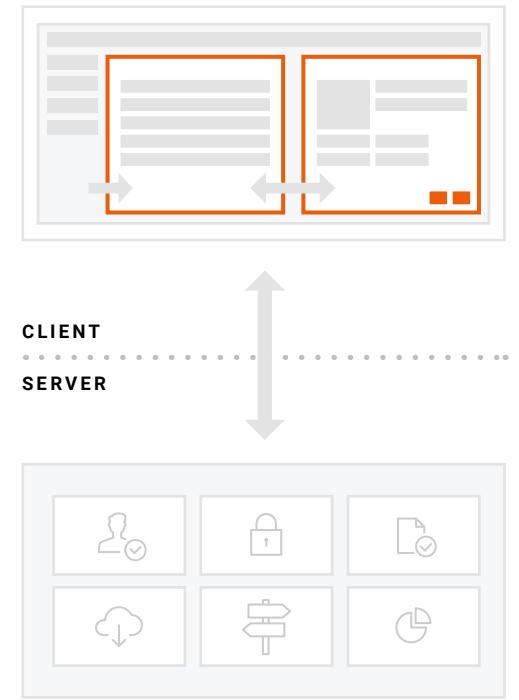
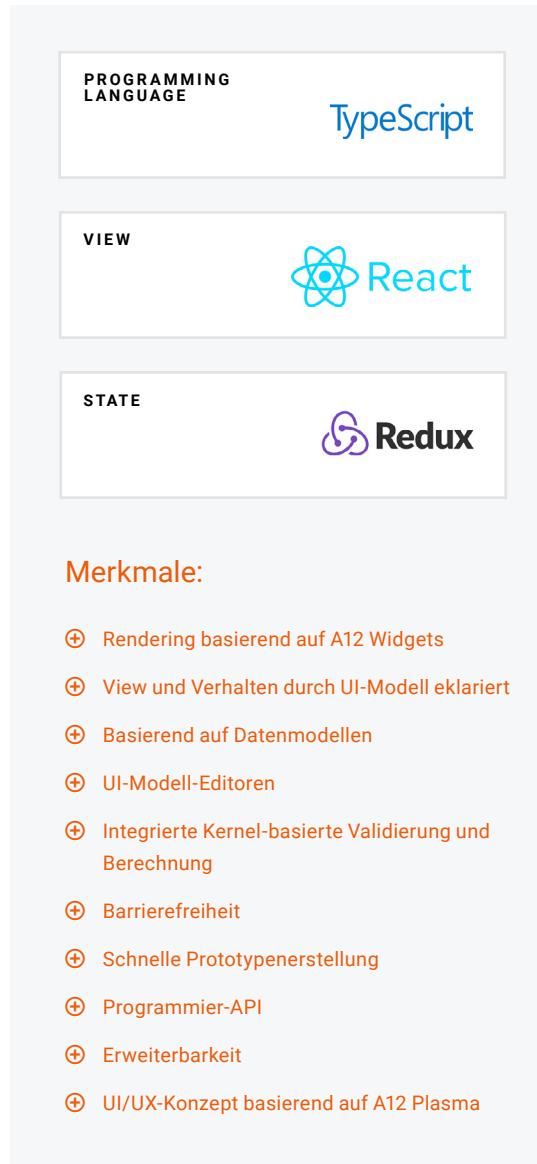
Widgets sind wiederverwendbare UI-Komponenten, die den Plasma-Design-Konventionen und UX-Konzepten folgen. Sie unterstützen Geschäftsanwendungen, die auf Desktops, Tablets und Smartphones mit Tastatur-, Maus und Touch-Eingabe laufen. Die Komponenten bieten eine einfach zu verwendende, gut dokumentierte, stark typisierte API und sind erweiterbar und anpassbar.



5. Laufzeitplattform

E Engines

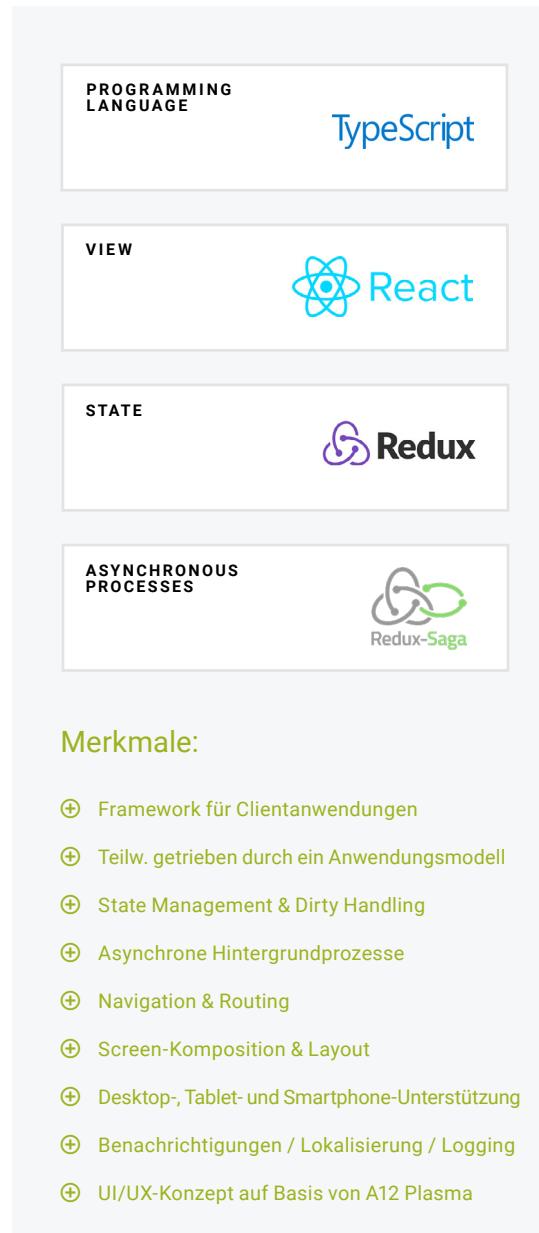
A12 Engines sind in Typescript implementierte, in sich geschlossene Laufzeitkomponenten, die Daten- und UI-Modelle interpretieren. Sie basieren auf den Plasma UI/UX-Konzepten und nutzen die Widgets für das Rendering.



5. Laufzeitplattform

C Client

Die modellgetriebene, Client-seitige Laufzeit-Komponente ermöglicht die Deklaration der Kernspekte der Anwendung, der Module, der Navigation, der Screens und der wichtigsten Interaktionsmuster. Ihre Hauptaufgabe ist die Orchestrierung anderer UI Komponenten, insbesondere der A12 Engines. Darüber hinaus organisiert sie die Bearbeitung von Anfragen, die Datenabfrage und -verarbeitung sowie die Zustandsverwaltung. Die Client-Komponente setzt das UI/UX-Konzept des Plasma Design Systems um und unterstützt Desktop, Tablet und Smartphone.



Merkmale:

- ⊕ Framework für Clientanwendungen
- ⊕ Teilw. getrieben durch ein Anwendungsmodell
- ⊕ State Management & Dirty Handling
- ⊕ Asynchrone Hintergrundprozesse
- ⊕ Navigation & Routing
- ⊕ Screen-Komposition & Layout
- ⊕ Desktop-, Tablet- und Smartphone-Unterstützung
- ⊕ Benachrichtigungen / Lokalisierung / Logging
- ⊕ UI/UX-Konzept auf Basis von A12 Plasma



5. Laufzeitplattform

K Kernel

Die Kernel-Komponente bündelt grundlegende Funktionen für die Erstellung und Verarbeitung von Datenmodellen. Sie definiert allen voran die domänen spezifische Sprache (engl. Domain specific language, kurz DSL) von A12. Dazu gehören sämtliche Grundlagen für Validierungen und Berechnungen im Rahmen der fachlichen Modellierung. Die Komponente enthält Client- und Serverseitige Laufzeitkomponenten sowie eine Java- und eine Typescript-API.

PROGRAMMING LANGUAGE

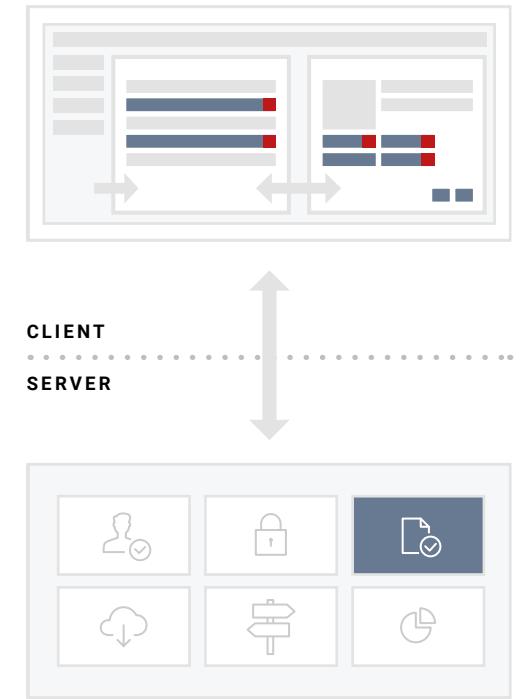


Java TypeScript

Die Kernel-Komponente beinhaltet u.a. die DSL von A12.

Merkmale:

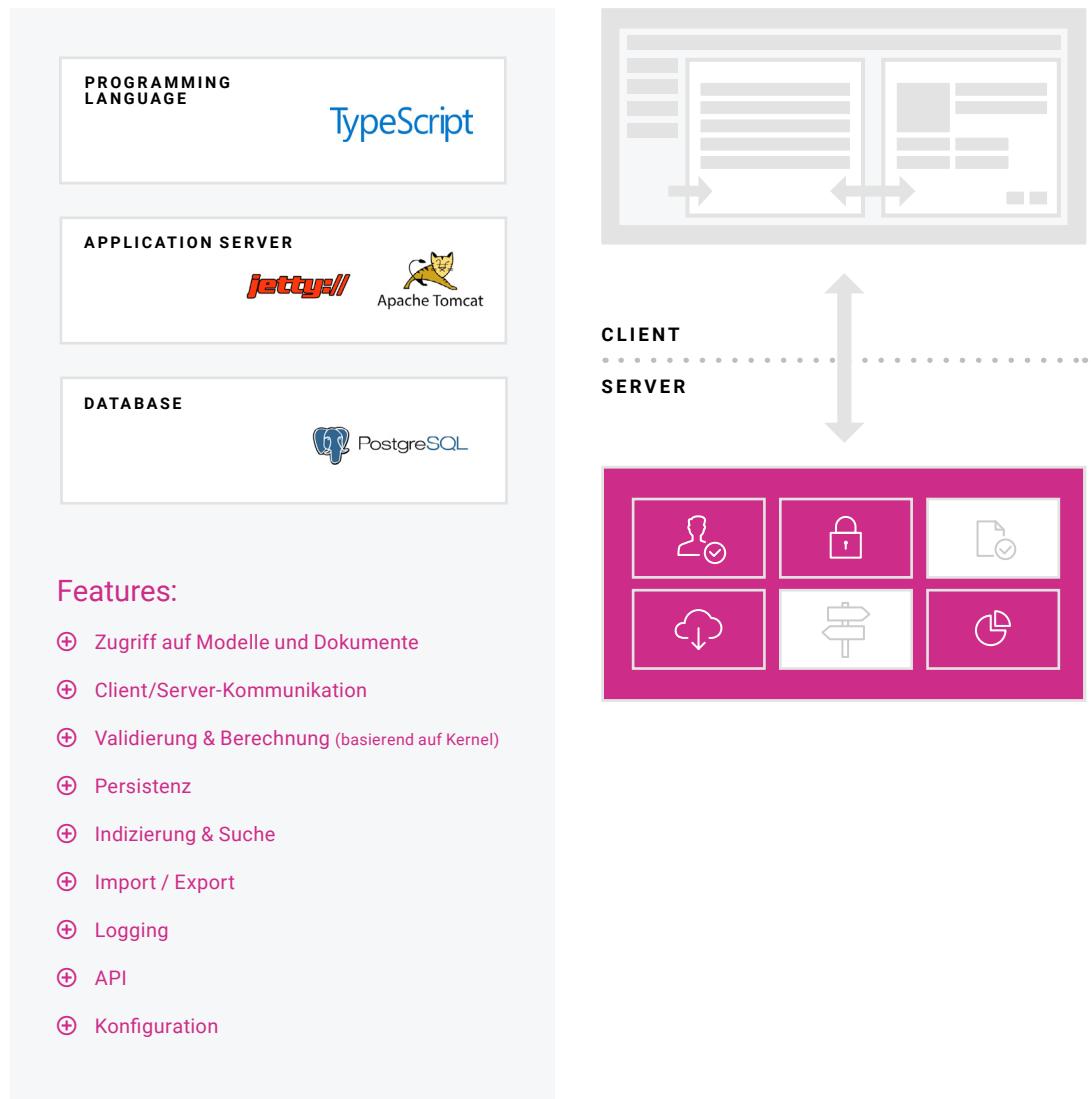
- ⊕ Angetrieben vom Datenmodell
- ⊕ Datenmodell-Editor
- ⊕ Feldtyp-Überprüfung
- ⊕ Feldtyp-Konvertierung
- ⊕ Validierung & Berechnung
- ⊕ Client- und serverseitige Laufzeitkomponenten
- ⊕ Programmier-API (Java & Typescript)



5. Laufzeitplattform

D Data Services

Die A12 Data Services-Komponente stellt eine API für die Verwaltung von Modellen bereit. Sie enthält außerdem Routinen für Client/Server Kommunikation, Authentifizierung, Autorisierung, Validierung, Persistenz, Indizierung. Sie wird in Typescript für die Client-Seite und in Java für Client- und Server-Seite zur Verfügung gestellt.



5. Laufzeitplattform

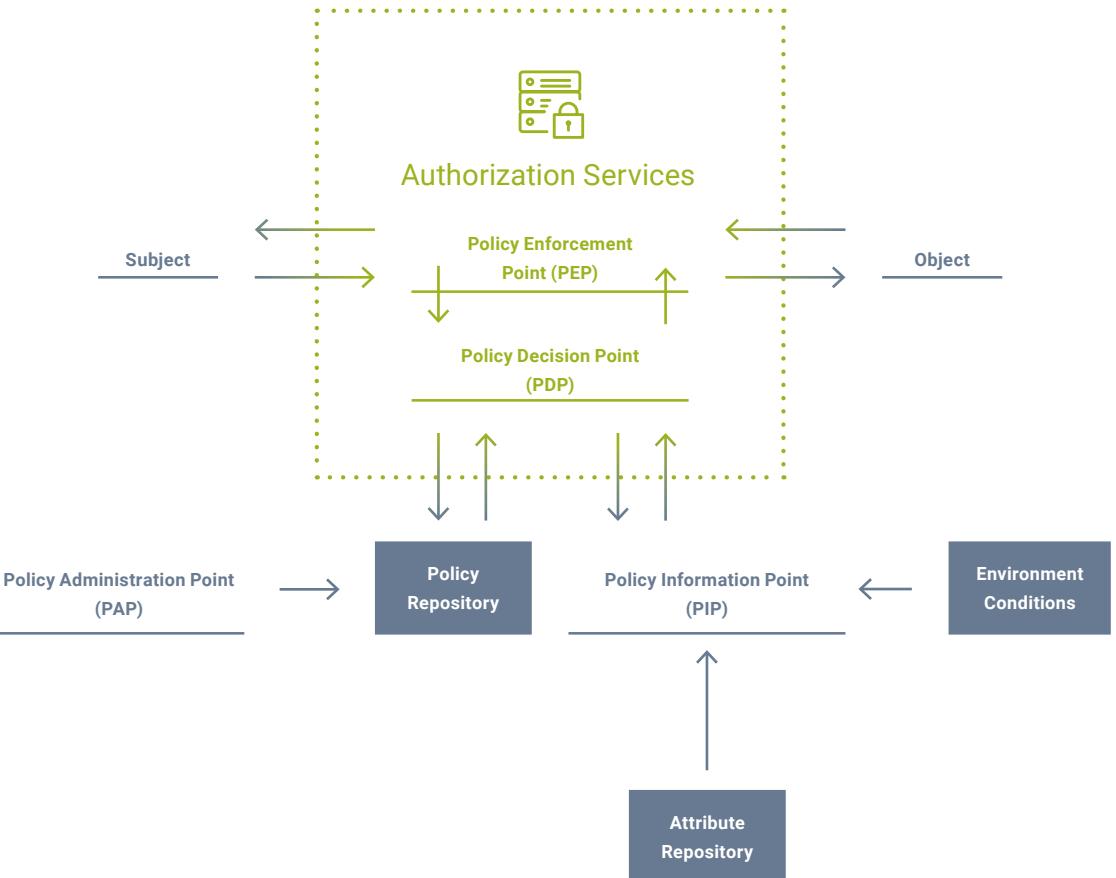
UAA User Management, Authentication & Authorization (UAA)

Für die Authentifikation wird in A12 die bewährte Open-Source-Lösung Keycloak eingesetzt. Es werden sowohl OAuth 2.0 mit OpenID als auch eine SAML-Token-basierte SSO-Authentifizierung und die Anbindung an LDAP unterstützt.

Die UAA Komponente von A12 bietet auch eine sehr flexible und leistungsfähige Autorisierungslösung, in der Zugriffsrechte in verschiedenen Granularitätsstufen möglich sind. So lassen sich sowohl rollenbasierte als auch komplexere, attributbasierte Regeln spezifizieren und damit Zugriffe bis auf Feld-Ebene von Datendokumenten schützen.

Die UAA-Komponente wird als Bibliothek bereitgestellt, sodass sie sowohl im A12 Server als auch in Standalone-Services der Anwendung eingebunden werden kann. Die Zugriffsregeln und weitere Autorisierungskonfigurationen werden von einem Policy-Repository bezogen.

Die UAA-Lösung richtet sich dabei nach der bekannten NiST ABAC-Referenzarchitektur.



Die UAA-Lösung richtet sich nach der NiST ABAC-Referenzarchitektur.

5. Laufzeitplattform

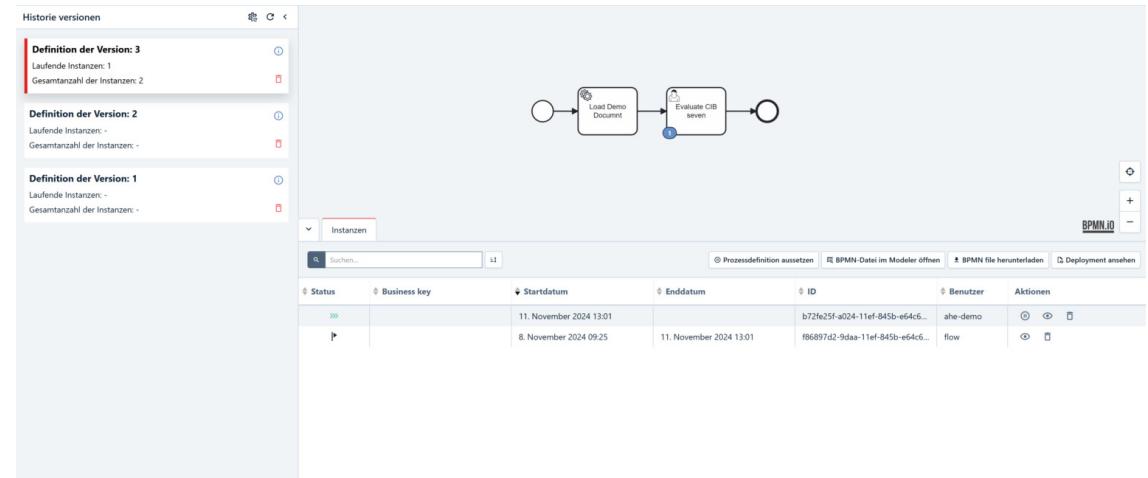
WF

Workflows

A12 Workflows bietet einen leichtgewichtigen Dienst, der BPMN-Modellierungsfunktionalität (Business Process Model and Notation) in A12 integriert und die grafische Modellierung serverseitiger Workflows und deren Ausführung ermöglicht.

Der A12-Workflow-Dienst kann als Erweiterung zu anderen A12-Produkten aktiviert werden und integriert sich nahtlos in die A12-Architektur. Auf diese Weise können Dokumente als Input und Output für einen A12-Workflow dienen und die Benutzeroberfläche für Benutzeraufgaben kann mit dem bestehenden A12-Modellierungsansatz erstellt werden.

Neben Benutzeraufgaben lassen sich auch automatisiert ausführbare Aufgaben wie Dienstleistungsaufgaben oder Skriptausführungen modellieren, wodurch sich auch teil- und vollautomatisierte Workflows unter Verwendung von Prozessbausteinen realisieren lassen. Als zentraler Bestandteil von A12 Workflows wird die BPMN Workflow Engine von CIB Seven verwendet.



Features:

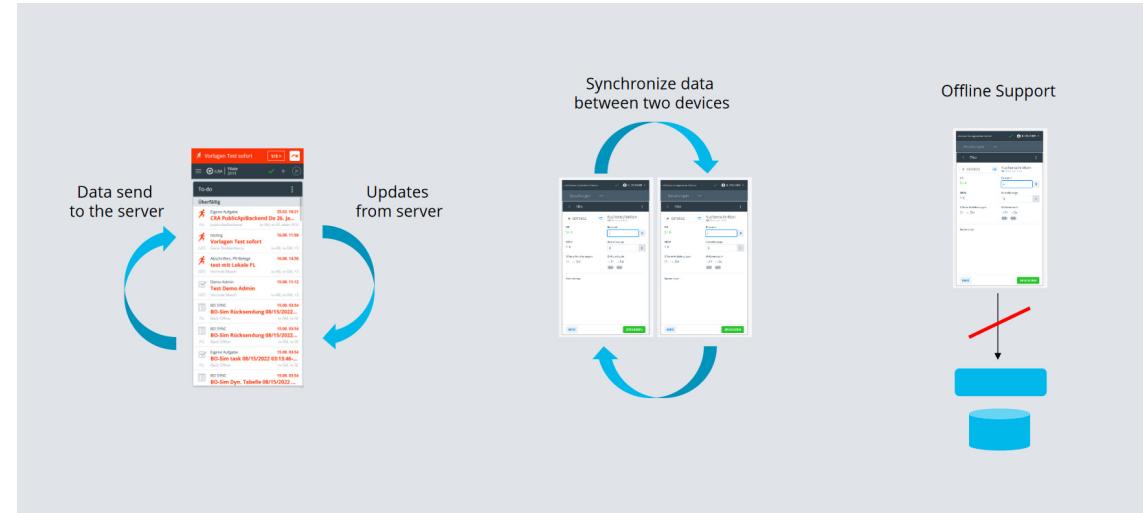
- ⊕ Modell-getriebene Geschäftsprozesse
- ⊕ Serverseitig / asynchron / semi-automatisiert
- ⊕ BPMN2
- ⊕ CIB Seven process engine
- ⊕ Camunda Modell-Editor
- ⊕ Integration mit A12-modellierten Daten

5. Laufzeitplattform

DD

Data Distribution

Die schnelle und sichere Synchronisation von Daten gehört in ausgewachsenen Geschäftsanwendungen zu den anspruchsvollsten technischen Aufgaben – vor allem dann, wenn nicht alle involvierten Systeme permanent online sind. Die Data Distribution-Komponente von A12 ist eine Transportschicht, die genau darauf spezialisiert ist. Der technische Dienst ist dafür ausgelegt, Daten zwischen Servern und Clients zu verteilen und Änderungen zu propagieren – insbesondere auch in Szenarien, in denen Clients temporär offline sind. Der Ursprung der Komponente liegt in einem E-Commerce-Projekt, in dem sie den Datenabgleich eines weltumspannenden Filialnetzes bewerkstellt.

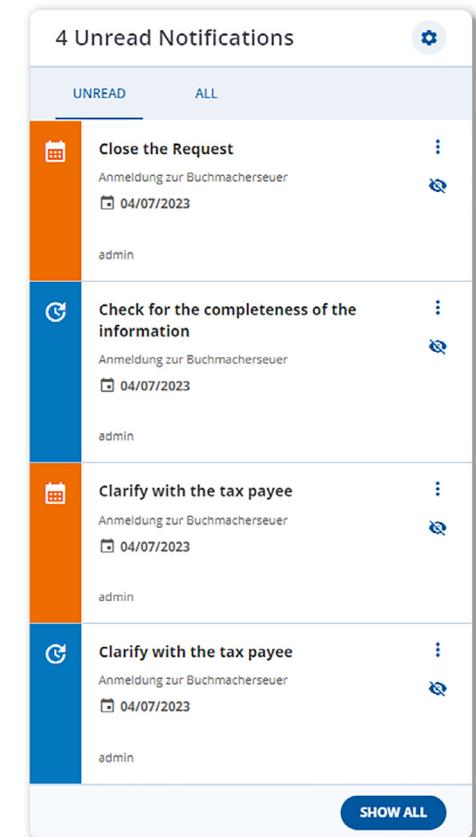
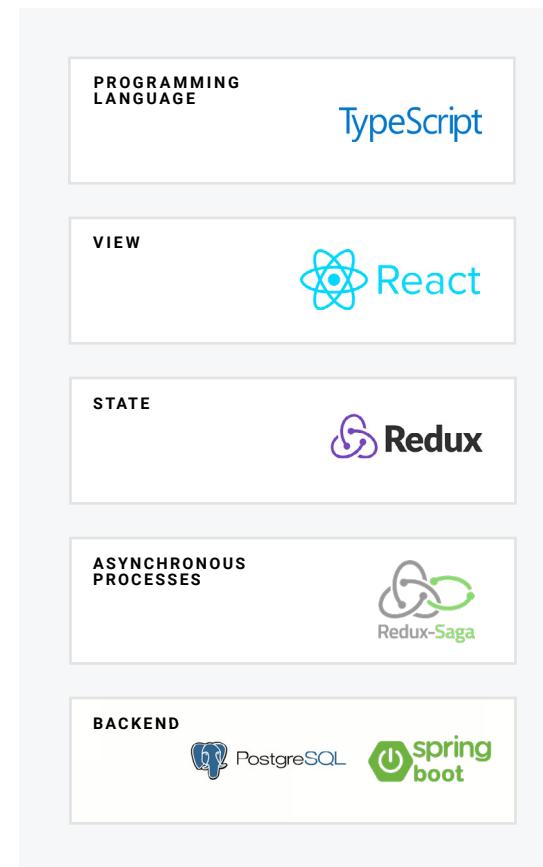


5. Laufzeitplattform

NC

Notification Center

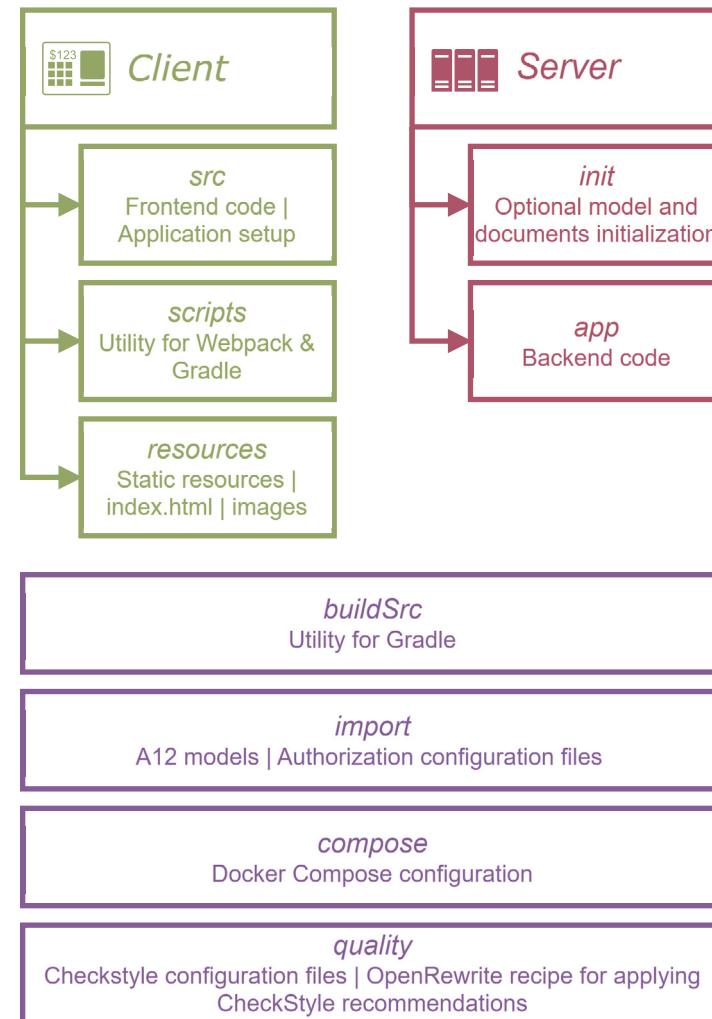
Im Rahmen von Geschäftsvorgängen strömen auf Mitarbeitende typischerweise eine Reihe unterschiedlicher Meldungen ein. Es gibt Informationen zu neuen Aufgaben, Nachrichten aus verschiedenen Kommunikationskanälen sowie Termine und Erinnerungen. Mit dem Notification Center lassen sich all diese Mitteilungen in Geschäftsanwendungen an einem zentralen Ort bündeln. Es dient als Sammelpunkt für verschiedene Arten von Benachrichtigungen basierend auf unterschiedlichen Business Use Cases, strukturierten Ansichten, verschiedenen Filtern und Benutzerpräferenzen. Das Notification Center lässt sich nahtlos in A12-basierte Anwendungen integrieren. Es stellt mehrere vordefinierte Benachrichtigungstypen bereit. Mit Hilfe der Programmbibliothek des Notification Centers kann das Entwicklungsteam auch eigene, individuelle Benachrichtigungstypen schnell und komfortabel erstellen.



5. Laufzeitplattform

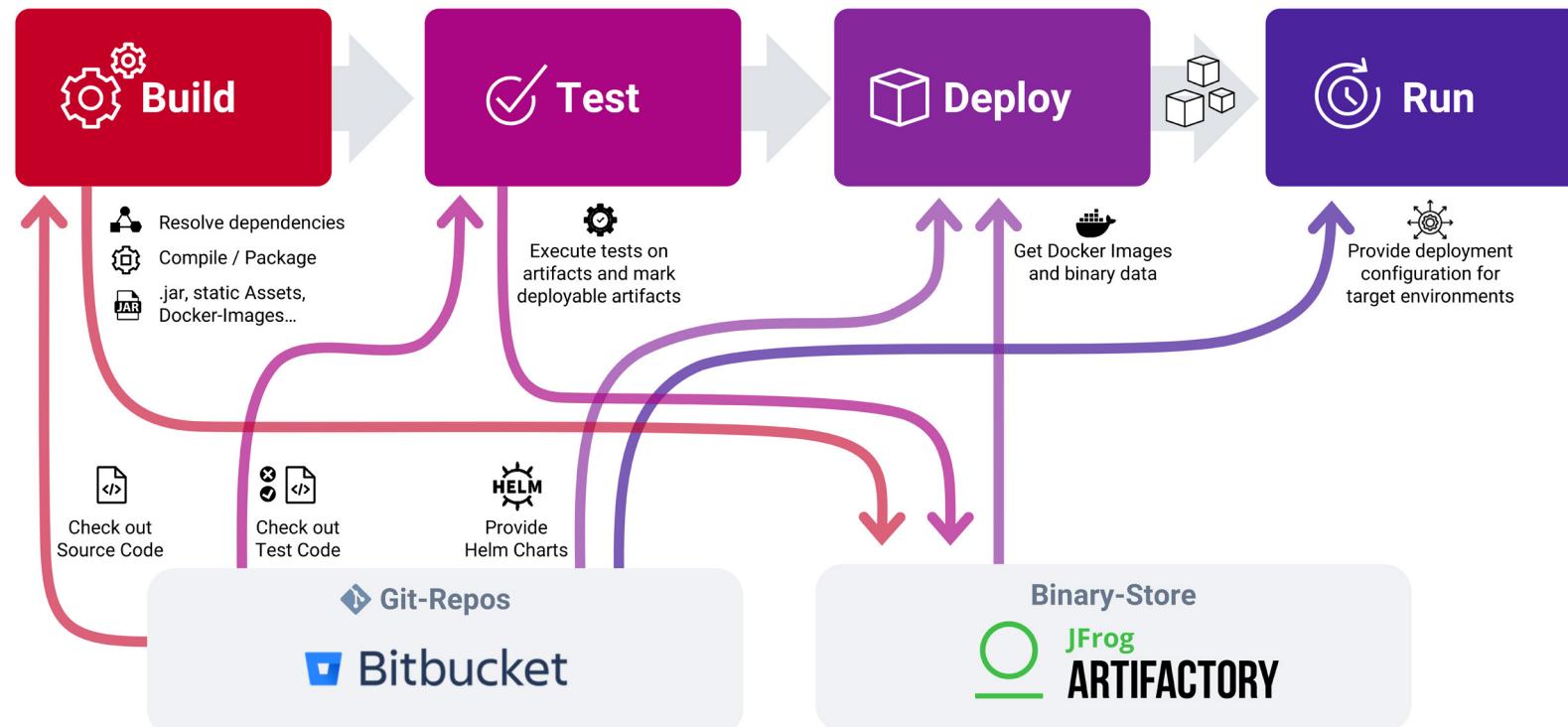
Projekt-Template

Mit dem A12 Projekt-Template steht für Entwicklungsteams ein Startpunkt bereit, um A12-Projekte komfortabel aufzusetzen und A12-Anwendungen schnell in Produktion zu bringen. Es enthält u.a. standardisierte Build-Pipelines sowie Entwicklungs- und Testumgebungen und deckt grundlegende Sicherheitsanforderungen ab. Als Kern beinhaltet das Template die A12 Komponenten Data Services, Client und UAA. Als Identity Provider ist Keycloak gesetzt, der Authentifizierungstyp ist im Standardfall OpenIDConnect/Oauth2. Optionale Komponenten wie Workflows, das Notification Center und die Print Engine lassen sich auf eine standardisierte Art und Weise einbinden.



5. Laufzeitplattform

Betrieb



Die A12-Plattform lässt sich On-Premise im unternehmens-eigenen oder externen Rechenzentrum betreiben. Darüber hinaus bietet mgm einen Betrieb in der Private Cloud von mgm an, gehostet in einem Rechenzentrum in Deutschland. Eine weitere Option ist der Cloud-Betrieb bei einem beliebi-gen Cloud-Anbieter.

Clusterfähigkeit – A12 ist Kubernetes-ready

A12-Anwendungen sind für das Deployment auf Kubernetes-Clustern ausgelegt. Aufbauend auf den Standards des A12 Projekt-Templates bietet A12 einen standardisierten Weg, um Anwendungen zu bauen und in alle gängigen Entwicklungs- und Produktions-Umgebungen (DEV, TEST, und PROD-Cluster) auszuspielen.

Operativ involviert der Betrieb die folgenden vier Schritte, in denen jeweils bewährte Standards zum Einsatz kommen.

Build – vom Quellcode zum Artefakt

Die Build-Phase wird zu unterschiedlichen Zeiten getriggert - zum Beispiel dann, wenn Entwickler ihre Codeänderungen in das Git-Repository des Projekts committen und pushen. Darüber hinaus berücksichtigen die A12 Templates für Build- & Deployment Pipelines auch Nightly-, Feature- und Release-Builds. Während des Build-Prozesses werden zunächst Abhängigkeiten aufgelöst, bevor der Code kompiliert

5. Laufzeitplattform

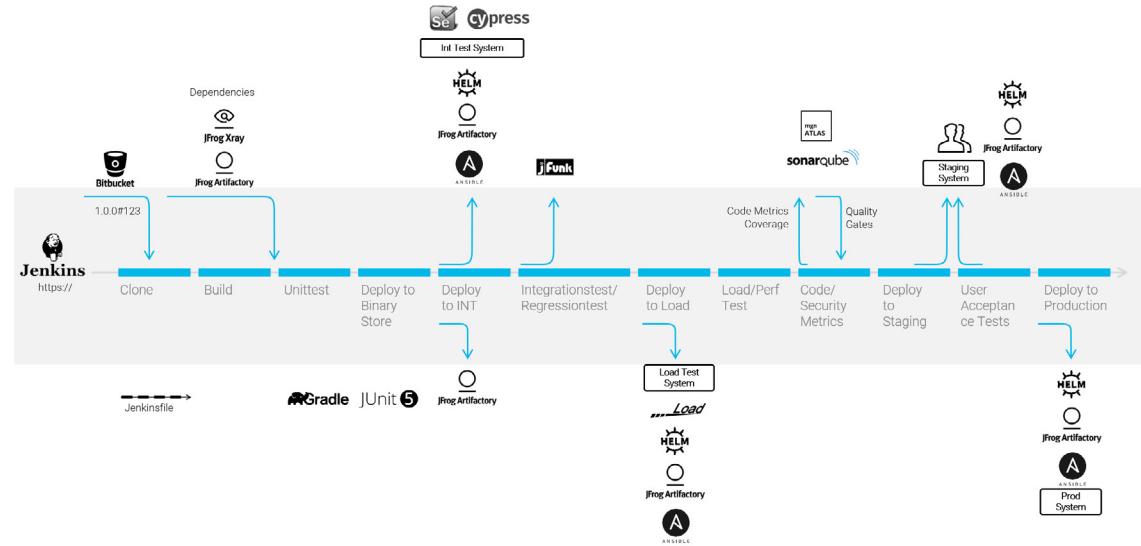
und paketiert wird. Die Build-Pipeline erstellt dann ein Docker-Image, das das Anwendungsartefakt zusammen mit seinen Laufzeitabhängigkeiten kapselt. Das resultierende Docker-Image und alle anderen Build-Outputs – wie JAR-Daten oder statische Assets – werden im Binary Store (mgm setzt standardmäßig auf Artifactory) veröffentlicht. Jeder Build wird mit detaillierten Metadaten versehen – darunter der Git-Commit-Hash, der Build-Zeitstempel, die Version und die Build-Nummer –, die jedes Artefakt in Artifactory direkt mit dem entsprechenden Quellcode in Git verknüpfen.

Test – Automatisierte Qualitätssicherung

Die **Test-Phase** stellt in mehreren Teilschritten sicher, dass nur qualitätsgesicherte Builds in der Pipeline weiter voranschreiten. Zu den Testarten gehören Unit-Tests, Integrationstests, End-to-End-Tests, Lasttests zur Ermittlung von Leistungsabfällen zwischen verschiedenen Versionen sowie Sicherheitstests. Der gesamte Testcode – von Unit- über Integrations- bis hin zu End-to-End-Tests – wird zusammen mit dem Anwendungscode in Git gespeichert, wodurch sichergestellt wird, dass die Tests mit der Anwendung synchronisiert bleiben. Die Tests werden mit den Artefakten durchgeführt, die in der Build-Phase erstellt wurden und aus Artifactory abgerufen werden. So ist die Konsistenz zwischen dem, was getestet wird, und dem, was letztendlich bereitgestellt wird, gewährleistet. Nur Artefakte, die alle obligatorischen Tests bestehen, werden als bereitstellungsfähig markiert.

Deploy – Sorgfältig konfigurierte Übergabe an die Zielumgebung

Die Deployment-Phase besteht im Wesentlichen aus zwei Schritten, und zwar der Festlegung der Konfiguration für den Betrieb der qualitätsgesicherten Builds in bestimmten



Zielumgebungen sowie der Übertragung in die jeweiligen Zielumgebungen. Für die Konfiguration steht der **Helm A12 Stack** bereit, der ein Kubernetes-Deployment out-of-the-box ermöglicht.

Neben einem Repository für den Programmcode enthält jedes A12-Projekt von Haus aus ein **Repository für die Konfiguration der Umgebungen**, in welche die Software deployed wird. Code und Konfiguration sind dadurch sauber voneinander getrennt. Außerdem triggern Änderungen an der Konfiguration automatisch bestimmte Jenkins-Jobs. So lässt sich zum Beispiel durch die Anpassung der Konfiguration eine bestimmte Version in die TEST-Umgebung aufspielen. Gleichzeitig ist stets transparent, wer wann welche Version aufgespielt hat.

Während des Deployments ruft die Pipeline die genehmigten Docker-Images und erforderlichen Binärdateien aus Artifac-

tory ab. Mit den Konfigurationsdaten aus Git steuern Jenkins-Skripte die Zielumgebungen – wie TEST, STAGE oder PROD – auf den entsprechenden Clustern an. Jedes Deployment wird sorgfältig protokolliert, wodurch eine lückenlose Rückverfolgbarkeit gewährleistet ist. Im Falle von Problemen können Deployments auf frühere Artefaktversionen zurückgesetzt werden, wobei die versionierte Speicherung von Artifactory und die umfassende Historie von Git genutzt werden.

Run – Verlässlicher Betrieb im Cluster

Die **Ausführungsphase** umfasst schließlich die laufende Überwachung, Wartung und Skalierung der Anwendungen in der Produktion. A12 setzt hier u.a. auf bewährte Standards beim **Logging & Monitoring**. Falls die jeweilige Betriebsumgebung keine bestimmten Logging- und Monitoring-Lösungen vorgibt, lässt sich das standardisierte Logging- und Mo-

5. Laufzeitplattform

nitoring-Setup von A12 (basierend auf Loki und Prometheus) nutzen. Der Zustand des Gesamtsystems kann jederzeit über Grafana Dashboards geprüft werden.

Wenn dringende Korrekturen oder Updates erforderlich sind, wird neuer Code in Git committet, wodurch automatisch die gesamte Build-, Test- und Bereitstellungspipeline ausgelöst wird, sodass aktualisierte Artefakte in Artifactory veröffentlicht und erneut bereitgestellt werden.

Hosting-Optionen für mehrere A12-Anwendungen

Für das Hosting von mehreren A12-basierten Anwendungen gibt es eine Reihe von Optionen:

- ⊕ **Isolation über Benutzerrechte und ansonsten »Mischbetrieb«**

Es ist möglich, eine zentrale A12 Plattform zu betreiben, auf der mehrere separate A12-Anwendungen laufen. Wenn zum Beispiel Benutzer auf mehrere Fachanwendungen zugreifen können, so könnte man die Daten- und Modell-Sicht über Rechte steuern. Über alle Anwendungen hinweg genutzte Services können auch gemeinsam genutzt werden.

- ⊕ **Isolation durch getrennten Betrieb**

Wenn die A12-Anwendungen isoliert voneinander betrieben werden sollen, müssen die A12-spezifischen Dienste für jede separate Laufzeitumgebung einzeln deployed werden.

Modularisierung bereitgestellter Artefakte

Der Frontend-seitige Teil einer A12-Anwendung kann als NPM-Package bereitgestellt werden. Die Modelle werden separat in den entsprechenden Servern bereitgestellt: das Workflow-Modell wird in CIB Seven installiert (oder aktualisiert) und die Daten/Formular-Modelle in den A12 Platform-Server (über die Import-API per REST-Aufruf) eingespielt.

Für die Kommunikation mit Umsystemen sind mehrere Optionen möglich:

- ⊕ **Daten aus Umsystem als A12 Documents dem Client verfügbar machen**

Variante 1: Umsystem pusht aktiv Daten

Das Bereitstellen der Daten aus dem Umsystem geht aktiv vom Umsystem aus, etwa per JMS Messaging (transaktional sicher) direkt in den A12 Server (der dazu um JMS-Listener erweitert wird). Oder indem die Data-Services APIs remote auf Seite des Umsystems aufgerufen werden. Hierzu bieten wir eine JSON-RPC API mit CRUD- und weiteren Operationen an. Diese Operationen können im Batch geschickt werden, die dann in einer gemeinsamen Transaktion verarbeitet werden. Aber man kann auch eigene Spring MVC REST-Endpoint oder JSON-RPC »Custom Operations« definieren – das wird in vielen Projekten erfolgreich praktiziert.

Variante 2: A12 Data Server ruft bei Bedarf Umsystem auf (»ersetzt Datenbank«)

Man kann leicht ein eigenes Spring Repository für einen Document-Typ implementieren, das die CRUD- und Listen-Operationen auf das Umsystem umleitet. Das Repository würde dann kein JDBC benutzen, sondern etwa per Messaging (JMS) oder REST/SOAP arbeiten. Zu beachten ist hier, dass nur JMS-Messaging in Java EE-Transaktionen laufen.

- ⊕ **Operationen des Umsystems »direkt« anbieten**

Wenn das Umsystem eher Operationen anbietet, bzw. die Daten direkt vom Client gesehen werden sollen (also nicht als A12-konforme Documents), dann kann man einen serverseitigen Service für den Client bereitstellen, der als Facade/Adapter zwischen Client und Umsystem dient. Als Standalone-Service kann dieser Service über Spring Boot bereitgestellt werden oder auf Basis eines anderen Frameworks oder einer Nicht-JVM Runtime. Die Authentifizierung und Authorisierung wird über das A12 UAA geleistet. Der Service kann die Aufrufe intern beliebig umsetzen. Als für den Client sichtbare Endpunkte wird REST oder besser JSON-RPC empfohlen.

- ⊕ **Direkter Aufruf der Umsysteme aus dem Client**

Dies ist zwar technisch möglich. Allerdings ist der direkte Zugriff auf Backend-Systeme ohne UAA wegen Sicherheitsbedenken und SSO/CORS Komplikationen nicht empfohlen.



Appendix A: Technologien

Durch die Trennung von Fachlichkeit und Technik lassen sich die eingesetzten Technologien bei Bedarf austauschen. Auf den Folgeseiten findet sich eine Übersicht des aktuellen A12 Technologie-Stack.

6. Appendix A: Technologien

Technologies currently in use

A12 PRODUCT	TECHNOLOGY	DESCRIPTION
K	Kernel	Java
		TypeScript
		Groovy
	Antlr	Parser generator
	StringTemplates	Template engine
	JAXB	Mapping Java objects to XML
	Jackson	JSON processor for Java
W	Widgets	TypeScript
		React Building UIs
		Styled Components CSS styling
	Recharts	Chart library
	DraftJS	Rich text editor
	React-Dnd	Drag and drop handling
	React-virtualized	Rendering partial data into DOM
	Redux	State management
UAA	UAA	TypeScript
		Redux State management
		oidc-client-js OpenIdConnect authentication protocol
	Java	
	Spring	Application framework for the Java platform
	Spring Boot	Auto configuration for Spring application

6. Appendix A: Technologien

	Spring-security	Spring security approach for authorization (SpEL - Spring Expression)
	KeyCloak	Identity and access management
	OAuth2/OpenID	Protocol for authentication
	SAML	Protocol for authentication
	LDAP	Protocol for accessing and maintaining distributed directory information services over an IP network
DS	Data Services	Java
		Apache Tomcat Application server
		Eclipse Jetty Application server
		PostgreSQL Database
		Spring Security Authentication, authorization
		Spring Boot Auto configuration for Spring application
		TypeScript API
WF	Workflows	Kotlin
		Spring Application framework for the Java platform
		Spring Boot Auto configuration for Spring application
		CIB Seven Platform for BPMN workflow and DMN decision automation
		TypeScript Frontend
		React Building UIs
		Webpack JavaScript module bundler
		NPM Package manager for JavaScript



6. Appendix A: Technologien

O	Overview Engine	TypeScript
	React	Building UIs
	React-Dnd	Drag and drop handling
	React-virtualized	Rendering partial data into DOM
	Redux	State management
F	Form Engine	TypeScript
	JavaScript	
	NodeJS	Java runtime environment
	NPM	Package manager for JavaScript
	Webpack	JavaScript module bundler
	React	Building UIs
	Redux	State management
	Marked	Markdown in expression language
	Jison	Expression language
	moment.js	JavaScript wrapper for the date object
T	Tree Engine	TypeScript
	React	Building UIs
	React-Dnd	Drag and drop handling
	React-virtualized	Rendering partial data into DOM
	Redux	State management

6. Appendix A: Technologien

C	Client	TypeScript	
		JavaScript	
		ESlint	Analysing TypeScript
		Prettier	
		NodeJS	Java runtime environment
		NPM	Package manager for JavaScript
		Lerna	Managing multi-package repositories
		Webpack	JavaScript module bundler
		React	Building UIs
		Redux	State management
		Inversify	Configuration injection
SME	Simple Model Editor	A12	Frontend
		TypeScript	
		React	Building UIs
		Redux	State management
		Redux Saga	Library used to handle side effects in Redux
Runtime	Docker / Docker-compose		Defining and running multi-container Docker applications
		Kubernetes	Managing containerized workloads and services
		Prometheus	Systems monitoring and alerting toolkit
		Grafana	Analytics & monitoring
		ELK (Elastic, Logstash, Kibana)	Log management
		Ansible	Automating configuration management & application deployment



mgm technology partners GmbH
Taunusstr. 23
80807 München
Tel +49 89 / 35 86 80-0
www.mgm-tp.com
info@mgm-tp.com



Innovation Implemented.