mgminsights

Innovation Implemented.

A12

Frequently Asked Questions (FAQ)

Impressum

mgm technology partners GmbH
Taunusstr. 23
80807 München
Tel +49 89 / 358 680-0

Gerichtsstand und Erfüllungsort: München

www.mgm-tp.com

Über dieses Dokument

Diese FAQ beantwortet eine Reihe häufig gestellter Fragen rund um die **Enterprise Low Code Plattform A12**. Eine strukturierte Einführung in A12 bietet das Whitepaper „*A12 – Low Code für individuelle Enterprise Software*“.

Der Fragenkatalog wird sukzessive erweitert. Haben Sie weitere Fragen? Kommen Sie gerne auf uns zu oder vereinbaren Sie gleich einen Demotermin.

Inhalt

1. Low Code

1.1. Was bedeutet „Low Code“? Was ist eine „Low Code Plattform“?

Das Buzzword Low Code bzw. Low Code Plattform wurde 2014 durch das Marktforschungsunternehmen Forrester ins Leben gerufen. Es bezeichnet eine Reihe ganz unterschiedlicher Ansätze, die in der Tradition modellbasierter Softwareentwicklung stehen. Eine Gemeinsamkeit ist das Prinzip, Programmcode auf Basis von Modellen zu generieren. Genau darauf bezieht sich der Begriff Low Code. Es geht darum, mit weniger manuell geschriebenem Code zu einer lauffähigen Software zu kommen. Der Softwareentwicklungsprozess soll dadurch beschleunigt werden, an Effizienz gewinnen und weniger voraussetzungsreich sein.

A12 gab es als modellbasierten Architekturansatz bereits, bevor das Buzzword Low Code auftauchte. Wir bezeichnen A12 heute nichtsdestotrotz als Enterprise Low Code Plattform. Denn der neue Begriff transportiert sehr gut die zentrale Idee, dass wir Software durch den Einsatz von Modellen und mittels Code-Generierung schneller und effizienter entwickeln sowie einfacher anpassbar und wartbar machen können.

1.2. Wie hängen modellbasierte Softwareentwicklung und Low Code zusammen?

Low Code ist ein neuer Begriff für Verfahren, die in ähnlicher Form seit Jahrzehnten erprobt werden. Dazu gehört insbesondere die modellbasierte Softwareentwicklung. Um Anwendungen mit weniger manuell geschriebenen Code zu realisieren, muss ein Teil des Codes generiert oder interpretiert werden. Die Voraussetzung ist eine vorangegangene Modellierung, typischerweise in einer domänenspezifischen Sprache.

1.3. Was ist eine „Enterprise Low Code Plattform“? Was versteht mgm darunter?

Bei den bestehenden Low Code Ansätzen gibt es große Unterschiede. Einige Anbieter stellen Plattformen als geschlossene Ökosysteme bereit, in denen Anwender kleine Apps zusammenklicken und veröffentlichen können. Dieses Baukastenprinzip hat den Vorteil, dass Ergebnisse sehr schnell sichtbar sind. Die Einarbeitung ist minimal, es sind kaum Vorkenntnisse erforderlich. Es hat allerdings auch den Nachteil, dass nur eine begrenzte Komplexität abgebildet werden kann. A12 ist hingegen als offene Plattform konzipiert, die sehr nah an dem modernen Werkzeugkasten eines Entwicklers bleibt. Je nach Problemstellung lassen sich flexibel unterschiedliche Lösungsansätze verwenden.

Wenn wir uns den Softwareentwicklungsprozess wie eine Fertigungsstraße vorstellen, dann erfordern unterschiedliche Stationen unterschiedliche Grade an Variabliität. Manchmal reichen vorgefertigte Bausteine, die eins zu eins wiederverwendet werden. Manchmal müssen wir die Bausteine parametrisieren und konfigurierbar machen, um die nötige Variabilität abbilden zu können. Wenn bestimmte Bausteine in sehr vielen unterschiedlichen Konfigurationen immer wieder benötigt werden, können wir domänenspezifische Sprachen entwickeln. Damit wird eine noch größere Variabilität beherrschbar. Dann gibt es aber auch immer wieder Fälle, die hochkomplex sind, aber nicht immer wieder auftauchen. Hier ist eine individuelle Umsetzung der beste Weg. Wir gehen davon aus, dass Projekte für geschäftskritische Enterprise Anwendungen immer auch Anteile enthalten, die indivuell entwickelt werden müssen. Und zwar, weil das in vielen Konstellationen schlichtweg der effizienteste und nachhaltigste Weg ist. Unnötige Abstraktionen führen zu einer unnötigen Komplexität und diversen Folgeproblemen wie einer erschwerten Wartbarkeit.

Als Enterprise Low Code Plattform bezeichnen wir einen problemadäquaten Ansatz, der die oben skizzierten Entwicklungsoptionen kombiniert. Low Code, wo möglich und sinnvoll. Individualentwicklung, wo nötig.

2. A12 Allgemein

2.1. Was ist A12?

A12 ist eine **Enterprise Low Code Plattform** für die Realisierung von Unternehmensanwendungen in komplexen IT-Landschaften.

Die **Modellierungsplattform** von A12 stellt Werkzeuge bereit, um Teile einer Anwendung ohne Programmierkenntnisse schnell zu erstellen und langfristig zu pflegen. Die **Laufzeitplattform** von A12 bietet die nötige Flexibilität, um Low Code Apps mit professioneller **Individualsoftwareentwicklung** und **Systemintegration** zu voll integrierten Unternehmensanwendungen zu entwickeln.

2.2. Wofür steht die Bezeichnung A12?

A12 steht für „Allianz 2012“. Ursprünglich war A12 eine Initiative von mgm für projektübergreifende Zusammenarbeit. Das Bekenntnis für die Allianz markiert gleichzeitig den Startpunkt für die Entwicklung der Plattform und den Fokus auf modellbasierte Softwareentwicklung. Obwohl die Plattform in Hinblick auf die Technik mehrere Evolutionsstufen durchlaufen hat, sind wir dem Namen treu geblieben.

2.3. An wen richtet sich A12?

A12 richtet sich an mittelgroße und große Unternehmen und Behörden, die individuelle Software benötigen. Mit A12 lassen sich schnell Prototypen und einfache Anwendungen realisieren, die zu komplexen, voll integrierten Unternehmensanwendungen wachsen können.

2.4. Für welche Arten von Anwendungen ist A12 ausgelegt?

Mit A12 lassen sich hochskalierbare, sichere und robuste Webanwendungen entwickeln. Dazu gehören beispielsweise Underwriting-Plattformen für Industrieversicherungen, Portale, Antragsmanagement-Systeme und Fachverfahren für den öffentlichen Sektor sowie Online-Shops, Marktplätze und integrierte Lösungen für den Online-, Filial- und Versandhandel. A12 spielt die besonderen Stärken des modellbasiertes Ansatzes vor allem in geschäftskritischen Bereichen aus, in denen Anwendungen lange Bestand haben, aber aufgrund fachlicher Weiterentwicklungen oder veränderter regulatorischer Vorgaben immer wieder angepasst werden müssen.

2.5. Wie unterscheidet sich A12 von anderen Low Code Plattformen?

A12 kombiniert einen Low Code Ansatz, bei dem Fachexperten eine Anwendung ohne Programmierkenntnisse erstellen können, mit professioneller Individualsoftwareentwicklung und Systemintegration.

Der Fokus von A12 liegt nicht auf leicht zusammenklickbaren Apps für den temporären Einsatz. A12 liefert vielmehr eine Antwort auf die Frage, wie Anwendungen langfristig zu voll integrierten, geschäftskritischen Unternehmenensanwendungen werden können.

Damit verfolgt mgm einen anderen Weg als Outsystems, Mendix und Co. A12 ist keine Platform as a Service (PaaS) Dienstleistung, in der einfache Anwendungen zusammengeklickt und deployt werden können. Es ist kein geschlosenes Ökosystem. Als Enterprise Low Code Plattform kommt A12 nur in professionellen, individuellen Softwareentwicklungsprojekten zum Einsatz.

Eine weitere Besonderheit besteht darin, dass A12-Projekte direkte Anforderungen an die A12-Basis stellen können - analog zu den Anforderungen an die eigene Projektsoftware. Damit sind die Projekte viel stärker einbezogen, sie können die Weiterentwicklung von A12 sehr direkt beeinflussen.

Mit dem Plasma Designsystem ist A12 auch im Bereich UI/UX speziell auf die Anforderungen ausgewachsener Geschäftsanwendungen ausgerichtet.

2.6. Welche Vorteile bringt A12?

Ein zentraler Vorteil von A12 ist die Trennung von Fachlichkeit und Technik. Vor allem geschäftskritische Software, die viele Jahre Bestand hat, profitiert davon enorm. Fachliche Inhalte, die einem ständigen Wandel unterliegen, können durch die Modellierung deutlich schneller und mit geringerem Aufwand in der Software abgebildet werden. Technische Innovationen können 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.

Selbstbestimmter Umgang mit fachlichen Inhalten
Fachexperten und Business Analysten können mit Modellierungswerkzeugen den fachlichen Kern der Software eigenständig abbilden und langfristig pflegen.

- Anpassung fachlicher Aspekte ohne Programmierkenntnisse
- Schnelle Umsetzung fachlicher Änderungen
- Weitgehende Automatisierung des Softwareentwicklungsprozesses
- Losgelöste Innovierung der Technik

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.

- Low Code, Individualsoftwareentwicklung und Systemintegration aus einem Guss
- Flexible Nutzung modular aufgebauter Laufzeitkomponenten
- Konsequenter Einsatz von Open Source Technologien
- Volle Kontrolle bzgl. Betrieb – On-Premise oder (Private) Cloud-Betrieb
- Möglichkeit, Anforderungen direkt an die A12-Basis zu stellen

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

2.7. Was bedeutet Co-Innovation?

Geschäftsprozesse greifen immer häufiger über die Unternehmensgrenzen hinaus. Sie docken an digitale Ökosysteme an. Der digitale Wandel hat das Tempo erhöht, mit dem Unternehmen auf neue Anforderungen und Marktbedingungen reagieren müssen. Der Innovationsdruck steigt.

Als Folge davon kann es sich heute kaum ein Unternehmen noch leisten, völlig eigenständig durch die Ungewissheiten des kontinuierlichen Wandels zu navigieren. Selbst Konkurrenten gehen Allianzen ein und teilen bestimmte Ressourcen. So ist es beispielsweise in der Automobilindustrie längst üblich, gemeinsame Investitionen in Plattformen, Komponenten und Produktionsketten zu tätigen. Kooperationsmodelle wie Co-Innovation, Open Innovation und Co-Creation finden immer weitere Verbreitung.

Genau dieses Prinzip liegt dem Community-Gedanken von A12 zugrunde. Selbst die individuellsten Kundenprojekte besitzen einen Kern gemeinsamer Anforderungen. Anstatt in isolierten Projektteams das Rad immer wieder neu zu erfinden, bündeln wir in A12 projektübergreifende Herausforderungen. Für den gemeinsamen Kern entwickeln wir robuste, skalierbare und langlebige Komponenten, die von den Effizienzgewinnen des modellbasierten Ansatzes profitieren. Dieser Kern ist für unsere Kunden eine geteilte Ressource im Sinne des Co-Innovation-Ansatzes.

2.8. Inwieweit bindet mich die Nutzung von A12 an mgm?

A12 ist als offenes System ausgelegt, das eine größtmögliche Flexibilität für die langfristige Pflege und Weiterentwicklung der Software ermöglicht. Es gibt keine Lock-in-Effekte. Einen Großteil der Anwendung - die modellierte Fachlichkeit - können Kunden von Anfang an eigenständig mitentwickeln und pflegen. Der Anteil individuell geschriebenen Codes ist im Vergleich zu klassischen individuellen Softwareprojekten deutlich geringer. Das vereinfacht auch die Übergaben - falls nach Projektabschluss ein internes Entwicklungsteam oder ein anderer Dienstleister die weitere Betreuung übernehmen soll.

2.9. Kann meine Abteilung A12 eigenständig nutzen?

Wenn im Rahmen eines Projekts eine A12-Anwendung erstellt wurde, ist die Fachabteilung in der Lage, die Anwendung durch Modellierungswerkzeuge eigenständig anzupassen. A12 ist jedoch kein App-Baukasten, der ein Zusammenklicken der Anwendung und ein Deployment auf Knopfdruck mit sich bringt. Eine Nutzung von A12 außerhalb eines Softwareentwicklungsprojekts ist in dieser Form nicht vorgesehen.

2.10. Gibt es optionale Add-Ons?

Ja, A12 unterstützt die Entwicklung zusätzlicher Add-Ons, die auf dem Kern von A12 aufsetzen. So gibt es zum Beispiel einen Code-Editor für den Parsergenerator ANTLR4 und eine Chat-Integration als Add-On. Damit lassen sich A12-basierte Applikationen sehr einfach um eine Chat-Funktionalität erweitern. Erweiterungen lassen sich zusätzlich zu anderen A12-Produkten aktivieren und können sowohl server- als auch clientseitige Komponenten umfassen. Der Chat-Bot-Service benutzt im Server beispielsweise das Open-Source-Produkt Rocket Chat. Das Chat-Frontend ist sowohl für die Verwendung an einem Desktop-PC als auch für Mobile Geräte optimiert.

3. Projektorganisation und -ablauf

3.1. Welche Formen der Zusammenarbeit sind bei einem A12-Projekt vorgesehen?

1. mgm übernimmt die gesamte Verantwortung für eine Anwendung.
   mgm nimmt die Anforderungen des Kunden im Rahmen einer Anforderungsanalyse und in laufenden Iterationen auf. Das mgm-Projekt-Team ist für die Erstellung der entsprechenden A12-Modelle und für alle Entwicklungs-, Test- und Release-Tätigkeiten verantwortlich.
2. Der Kunde übernimmt die fachliche Modellierung. mgm verantwortet die Entwicklung und Technik.
   mgm schult Fachexperten/Business Analysten des Kunden in der Erstellung von A12-Modellen. Der Kunde übernimmt im Projekt die fachliche Modellierung. Das ist eine natürliche und sinnvolle Aufgabenteilung, weil die Modelle immer die Fachlichkeit der Geschäftsdomäne abbilden. Darin kennt sich der Kunde viel besser aus, als mgm-Kollegen. mgm bietet Support bei Modellierungsfragen und verantwortet die gesamten Entwicklungs- und Technik-Aufgaben.

3.2. Was ist die mgm Projekt-Basis?

Unter der Klammer Projekt-Basis hat mgm eine Reihe von Werkzeugen, Standards und organisatorischen Prinzipien zusammengefasst, die für eine effiziente Projektabwicklung mit A12 ausgelegt sind. Dazu gehören durchdachte, integrierte Lösungen für alle entwicklungsbegleitenden und produktionsrelevanten Faktoren hochqualitativer Geschäftsanwendungen:von der Infrastruktur über QA, Security und Datenschutz bis hin zu Dokumentation, Betrieb und Schulung.

4. Security

4.1. Wie sicher ist A12? Welche Security-Merkmale weist A12 auf?

A12 folgt dem Grundsatz Security by Design. Sicherheitsanforderungen werden von Anfang an berücksichtigt, um potenziellen Schwachstellen präventiv vorzubeugen. Security-Experten begleiten alle Phasen der Entwicklung – von frühen Anforderungen über Architekturentscheidungen bis hin zu Abnahmetests.

Darüber hinaus wird die Enterprise Low Code Plattform kontinuierlich intensiv getestet. A12 nutzt dafür das von mgm entwickeltes Security Toolset ATLAS, das eine Reihe von Werkzeugen wie OWASP Dependency Check, ZAP und sqlmap integriert. Es ermöglicht automatisierte Security Tests und stellt ein konsolidiertes Reporting bereit. ATLAS prüft u.a. auf bekannte Schwachstellen in Komponenten von Drittanbietern, erkennt Konfigurationsprobleme wie fehlende HTTP Security Header und testet, wie robust APIs gegenüber Attacken wie Injektionsangriffen sind.

5. UI / UX

5.1. Was ist A12 Plasma?

A12 Plasma ist ein Designsystem, das mgm speziell für Geschäftsanwendungen entwickelt hat. Es besteht aus UI/UX-Komponenten, Verwendungsmustern und Gestaltungsrichtlinien, mit denen sich konsistente, effiziente und ansprechende Benutzeroberflächen gestalten lassen.

Der A12 Widget Showcase enthält Beispiele für alle verfügbaren Plasma Komponenten.

Im Gegensatz zu reinen Designsprachen wie Material Design berücksichtigt Plasma auch einen erweiterten Funktionsumfang, der typischerweise von Geschäftsanwendungen abverlangt wird. Dazu gehören insbesondere Aspekte wie Skalierbarkeit und der Umgang mit einer hohen Informationsdichte.

5.2. Wie funktioniert in A12 die Oberflächengestaltung?

A12 setzt bei der Gestaltung der Benutzeroberfläche auf spezielle Oberflächenmodelle. Sie setzen ebenfalls die Idee einer Trennung der Technik um. Sie ermöglichen eine abstrahierte Darstellung der Interaktionsstruktur – zum Beispiel den Aufbau eines Formulars – ohne dabei mit einer bestimmten technischen Umsetzung fest verdrahtet zu sein. Das hat den Vorteil, dass die technischen Darstellungsdetails und das Design losgelöst von den Oberflächenmodellen weiterentwickelt werden können. So lässt sich viel einfacher eine flächendeckend konsistente und barrierefreie Benutzeroberfläche realisieren. Für die Darstellung kommt das Plasma Designsystem zum Einsatz.

6. Modellierung

6.1. Welche Modellierungswerkzeuge gibt es?

Aktuell stellt A12 zwei Modellierungswerkzeuge bereit: den den A12 Simple Model Editor (SME) und den A12 UI Designer. Der Funktionsumfang des A12 UI Designers wird perspektivisch in den SME überführt, um sämtliche Modellierungsfunktionalität in einem Tool zu bündeln.

	A12 SIMPLE MODEL EDITOR	A12 UI DESIGNER
Kurzbeschreibung	Webbasiertes Modellierungstool der neuen Generation, das langfristig den gesamten Modellierungsumfang von A12 übernimmt	Desktop-Anwendung einer älteren Generation für die Definition von UI Modellen
Modellierungsumfang	App Model, Relationsships Model, Overview Model, Tree Model, Document Model, Regelsprache für Validierung und Berechnung	Form Model
Dateityp	JSON	JSON

6.2. Welche A12-Modelle gibt es?

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.
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 Modellierungswerkzeuge 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	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	Mit dem Print Model lassen sich Druckvorlagen für die Generierung barrierefreier PDFs erstellen.

6.3. Was kann ich mit A12 modellieren und was nicht?

Der aktuelle Modellierungsumfang umfasst die gesamte Fachlichkeit und Teile der Anwendungsoberfläche. Die Entscheidung, was in A12 modellierbar wird, richten wir an dem Mehrwert aus, den die Modellierbarkeit bringen würde. In komplexen Geschäftsanwendungen gibt es immer wieder Aspekte, die sich individuell schneller, effizienter und nachhaltiger entwickeln lassen.

Mit den Datenmodellen und der Regelsprache für Validierungen und Berechnungen lassen sich sehr komplexe fachliche Zusammenhänge abbilden. Geschulte Fachexperten und Business Analysten sind dadurch in der Lage, mit den Modellierungswerkzeugen die fachlichen Aspekte einer Anwendung eigenständig festzulegen - ohne dabei auf Entwickler angewiesen zu sein. Nur in Ausnahmefällen - zum Beispiel bei Berechnungen mit sehr komplexen Formeln - kann es komfortabler sein, die Berechnungen direkt im Code und nicht über die Modellierungstools abzubilden.

Die Modellierung der Oberfläche beschränkt sich derzeit auf die Bereiche, in denen modellgetriebene Komponenten zum Einsatz kommen.

MODELLIERBAR	INDIVIDUELL UMSETZBAR
Fachlichkeit - Datenmodelle mit Validierungsregeln und Berechnungen	komplexe Algorithmen (z.B. generischer Prämienrechner im Versicherungsumfeld)
Rahmen einer Anwendung inkl. Platzierung von modellgetriebenen Engines	Anwendungsmodell umfasst nicht die Platzierung einfacher Widgets
Formulare, inkl. wiederholbaren Strukturen	Keine Definition oder Anpassung von Designelementen
Tabellarische Übersichten von Datensätzen	
Baumartige Übersichten von Datensätzen	
Beziehungen zwischen verschiedenen modellgetriebenen Komponenten	

6.4. Gibt es ein Schulungsprogramm von mgm für die Modellierungstools?

Ja, das Business Professional Services Team von A12 bietet Trainings für den Umgang mit den Modellierungstools. Sie richten sich in erster Linie an Business Analysten, die in Projekten Teile der Modellierung übernehmen. Neben Präsenzschulungen, die je nach Vorwissen individuell gestaltbar sind, steht ein E-Learning Modul für die Einführung in die Datenmodellierung bereit.

6.5. Lassen sich A12-Modelle wiederverwenden?

Ja. Die Wiederverwendung ist bei allen Modelltypen möglich. Vor allem bei den A12-Datenmodellen ist sie gängige Praxis. Datenmodelle lassen sich modular aufbauen. So können untergeordnete Submodelle zum Beispiel in mehreren anderen Modellen wiederverwendet werden. Darüber hinaus können spezielle Typdefinitionen – zum Beispiel für zentrale Länderlisten und Rechtsformen – erstellt werden, die in mehreren Modellen genutzt werden können. So lassen sich modellübergreifende Aspekte an einer einzigen Stelle definieren, um Dopplungen und etwaige Inkonsistenzen zu vermeiden.

7. Betrieb

7.1. Welche Optionen bietet mgm für den Betrieb von A12-Anwendungen?

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 Rechenzentrum
- Betrieb in der Private Cloud von mgm, gehostet in einem Rechenzentren in Deutschland
- Cloud-Betrieb bei einem beliebigen Cloud-Anbieter

7.2. Unterstützt A12 Kubernetes?

Ja, A12-Anwendungen sind für das Deployment auf Kubernetes-Clustern ausgelegt. Basierend auf den Erfahrungen aus mehreren großen Softwareprojekten haben wir eine Auswahl an Tools aus dem Kubernetes Ökosystem getroffen, die wir als Standard-Stack empfehlen. Grundsätzlich lassen sich A12-Anwendungen jedoch mit unterschiedlichen Technologie-Stacks betreiben – je nach den Vorgaben des jeweiligen Hosters.

8. Technik

8.1. Lassen sich Teile der A12 Plattform auch einzeln verwenden?

Ja. A12 ist modular aufgebaut und in verschiedene Produkte gegliedert. Der Schnitt der A12 Produkte ist technisch motiviert. Jedes Produkt verfügt über einen klaren Scope und klare Schnittstellen nach außen. Die Produkte lassen sich flexibel einsetzen - auch einzeln. So kann man beispielsweise den Client verwenden und den Server selber schreiben.

8.2. Welche A12 Module / Produkte gibt es?

PRODUKTNAME	KURZFORM	BESCHREIBUNG
Client	C	Modellgetriebene, Client-seitige 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 Dokumentenmodellen: Modellierungswerkzeuge, Sprache für Validierungen und Berechnungen, Client- und Server-seitige 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
Simple Model Editor	SME	Modellierungswerkzeug für Business Analysten
Installer		Stellt alle aktuellen und aufeinander abgestimmten A12 Produkte und Tools in einem vorkonfigurierten Paket zur lokalen Installation bereit - damit kommen Business Analysten an eine Modellierungs- und Demoumgebung

8.3. Wo finde ich den Source Code von A12?

Der Quellcode von A12 ist aktuell nur für Entwicklungspartner in ausgewählten Großprojekten zugänglich.

8.4. Auf welchen konkreten Technologien basiert A12?

Durch die Trennung von Fachlichkeit und Technik lassen sich die eingesetzten Technologien bei Bedarf austauschen. Aktuell ist der Technologie-Stack von A12 folgendermaßen zusammengesetzt:

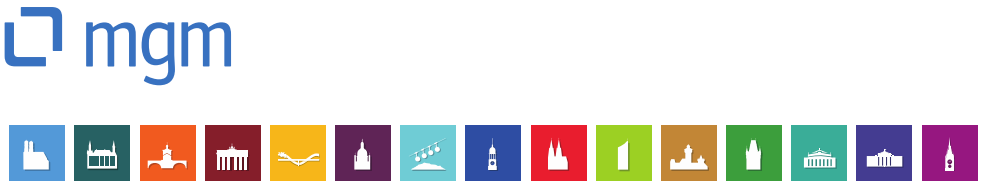
A12 PRODUCT	TECHNOLOGY	DESCRIPTION
Kernel	Java	
	Typescript	
	Groovy	
	Antlr	Parser generator
	StringTemplates	Template Engine
	JAXB	Mapping Java objects to XML
	Jackson	JSON processor for Java
Widgets	Typescript	
	React	Building UIs
	Stylus	CSS preprocessor
	Recharts	Chart library
	DraftJS	Rich text editor
	React-Dnd	Drag and drop handling
	React-virtualized	Rendering partial data into DOM
	Redux	State management
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
	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

Services	Java	
	Apache solr	Search index
	WildFly	Application server
	Apache Tomcat	Application server
	Eclipse Jetty	Application server
	PostgreSQL	Database
	Oracle	Database
	H2	Local In-Memory-DB
	Spring Security	authentication, authorization
	Spring Boot	Auto configuration for Spring application
	NodeJS	Java runtime environment
	Typescript API	
Workflows	Kotlin	
	Spring	Application framework for the Java platform
	Spring Boot	Auto configuration for Spring application
	Camunda	Platform for BPMN workflow and DMN decision automation
	Typescript	Frontend
	React	Building UIs
	Webpack	JavaScript module bundler
	NPM	package manager for JavaScript
Overview Engine	Typescript	
	React	Building UIs
	Stylus	CSS preprocessor
	Recharts	Chart library
	DraftJS	Rich text editor
	React-Dnd	Drag and drop handling
	React-virtualized	Rendering partial data into DOM
	Redux	State management
Form Engine	TypeScript	
	JavaScript	
	TSLint	Analysing Typescript
	NodeJS	Java runtime environment
	NPM	package manager for JavaScript
	Lerna	Managing multi-package repositories
	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
Tree Engine	Typescript	
	React	Building UIs
	Stylus	CSS preprocessor
	Recharts	Chart library
	DraftJS	Rich text editor
	React-Dnd	Drag and drop handling
	React-virtualized	Rendering partial data into DOM
	Redux	State management
Chat Solution	A12 BAP Client	frontend
	A12 Widgets	frontend
	Rocket.Chat	Web chat platform
	NodeJS	Java runtime environment
	MongoDB	Data persistence
Chatbot	Python	
	Rasa	Chatbot development framework
	Tensor-flow	Machine learning/differentiable programming framework
	Scikit-learn	Machine learning library
	Flask	Web framework
Client	Typescript	
	JavaScript	
	TSLint	Analysing Typescript
	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
Data Modeler	Java	
	Tycho	Building Eclilpse plugins
	RCP	Building Eclilpse plugins
	SWT	Widget toolkit for Java
	JFace	UI toolkit
	Jackson	JSON processor for Java
	JSONSchema	Validating the structure of json data
	Slf4J	simple facade or abstraction for various logging frameworks

	LOGBack	logging framework for Java applications
UI Designer	Java	
	Tycho	Building Eclipse plugins
	RCP	Building Eclipse plugins
	SWT	Widget toolkit for Java
	JFace	UI toolkit
	Jackson	JSON processor for Java
	JSONSchema	Validating the structure of json data
	Slf4J	simple facade or abstraction for various logging frameworks
	LOGBack	logging framework for Java applications
Simple Model Editor	A12	Front end
	Typescript	
	React	Building UIs
	Redux	State management
	Redux Saga	library used to handle side effects in Redux
A12 Installer	Typescript	
	React	Building UIs
	Redux	State management
	Redux Saga	library used to handle side effects in Redux
	Spring Boot	Auto configuration for Spring application
	H2 Database	Local In-Memory-DB
	Electron	Software framework to develop desktop GUI applications using web technologies
Plasma Design	Adobe illustrator	Creating graphical user interfaces
	Adobe XD	Creating screens and lo-fi prototypes
	Azure	Creating hi-fi prototypes
	PUG	Template engine – create reusable HTML
	BEM	Creating extendable and reusable CSS
Documentation	Asciidoc	User documentation
	Typedoc	Generating API documentation for TypeScript
	Javadoc	Generating API documentation for Java
QA, Testing & Security	Enzyme	Unit tests
	Cypress	Integration tests
	Testcontainers	Integration/system tests based on Docker containers
	JUnit 5	testing framework for java applications
	MockK	For Kotlin

	H2	Local In-Memory-DB
	QFS-Test-Suite	Automated surface tests
	PerfLoad	Load testing
	Selenium	Browser automation
	Mocha	JavaScript test framework
	TestCafe	Automating end-to-end web testing
	Sonarqube	Continuous inspection of code quality
	OWASP Dependency Check	Scanning for vulnerabilities
	TestRail	Managing and tracking testing
	JAX-RS	Integration tests
	jMeter	Functional behavior and performance tests
	TestNG	Unit, functional, end-to-end, integration tests
	Python	Orchestrating Security Test Suite
	Docker	Running Security Test Suite
	Sqlite, MariaDB	Persistent Storage for Licenses, Credentials, Configuration
	OWASP ZAP	Dynamic Application Security Testing
	Postman/Newman	REST client for API Testing
	OWASP DefectDojo	Security Reporting and Monitoring
	Xanitizer	Static Application Security Testing
	Chai	Assertion library for Node
	NYC	Test coverage reporting
	NPM audit	Security review of project's dependency tree
	Hamcrest	creating customized assertion matchers
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
Development-Infrastructure	Jenkins	Automation of builds and deployment
	Artifactory	Managing code repositories
	GIT	Version control
	Bitbucket	Code Collaboration & Version Control
	Gradle	Build automation
	Maven	Build automation
	Webpack	JavaScript module bundler
	NPM	package manager for JavaScript



mgm technology partners
www.mgm-tp.com

mgm consulting partners
www.mgm-cp.com

mgm security partners
www.mgm-sp.com