A12 - A partner community approach to build and run business software
We bring applications into production!

For over 25 years mgm has been developing web applications for Commerce, Insurance and E-Government: Highly scalable, secure, robust.

More than 700 colleagues represent our mission: **Innovation Implemented.**
Contents
Introduction

Tools
- A12 Data Modeler
- A12 UI Designer
- A12 Validation Language

Architecture
- Overview
- Client
- CAT/CAF
- Server

Concepts
- How A12 works – From models to applications
- A12 project workflow and development
- Data first – The modeling philosophy of A12
- Plasma Design – UI/UX for business applications

Building Blocks
- Widgets
- Engines
- App Templates
- Business Apps
Introduction
A12 is an extendable development platform for web based business applications. It provides developers with a rich and robust set of secure, scalable and operations-proof components and client/server application infrastructure. A12 enables business analysts to define major parts of the application via models using a set of convenient tools.
Customers benefit from:

- Robust high-quality software and a short time-to-market
- Synergies and knowledge transfer across projects
- Future-proof applications that can adapt to changing business needs
- New features coming from other projects and mgm's R&D efforts

mgm provides:

- A constantly refined state-of-the-art technical environment
- A growing set of A12 tools, components and best practices
- Well-established processes between developers, analysts and UI/UX experts
- A dedicated expert team and the results of many years of R&D

Share assets across projects
Enable analysts to create applications

Highlights
Create and adapt applications rapidly. Reuse business logic and UI patterns. Reduce custom development.

Build business apps
Rely on a model-oriented architecture
Concepts
Digitization brings along profound changes in all industries – from reorganized business structures to more demanding customer expectations. In terms of software, digital business calls for highly adaptive applications and robust implementations in short timeframes. The following articles sketch out how A12 speeds up application development and helps to seize new opportunities in digital business.
How A12 works – From models to applications
mgm A12 is a model-driven approach to business software. It provides a set of concepts, components and tools for creating modern, document-oriented web applications.

Data first – the modeling philosophy of A12
A12 is all about models. But what exactly can you model as a business expert? What are the benefits of having different models for data and the user interface? And how does A12 differ from GUI builders?

A12 project workflow and development
Speed up the creation of web applications and reduce custom development efforts: How are these objectives reflected in the workflow? And what about individual requirements?

Plasma Design – UI/UX for business applications
Interface and interaction design usually take up a lot of time and effort in individual software projects. Plasma Design by mgm aims at speeding up the process. It focusses specifically on business applications.
How A12 works – from models to applications

mgm A12 is a model-driven approach to business software. It provides a set of concepts, components and tools for creating modern, document-oriented web applications.

Decouple domain-specific information from development

The core idea of mgm A12 is to encapsulate domain-specific knowledge in models. By using a set of powerful tools, domain experts and business analysts are able to create and modify these models – without the need to touch any code. This concept significantly reduces custom development efforts. Moreover, it enables domain-experts to adapt their applications rapidly – which is a competitive factor in a digitized world with fast changing business requirements.

Focus on documents and forms

Most business transactions are handled via some kind of documents. This includes contracts, purchase orders and different kinds of requests for example. When it comes to digitizing documents, online forms play a vital role. They specify the structure of documents and determine which data is needed for the business transaction. In all industries, user data is typically retrieved via forms on the one hand. On the other hand, forms can be used as a kind of blueprint for displaying data to users. The A12 tools and components have a strong focus on modeling business processes and documents using a form-based approach. They provide the means to create complex forms with up to thousands of data fields, dependencies and validation rules and bring them into web applications.

Enable cross-platform and cross-device applications

In terms of business applications, the client-side has gained the most traction during the last years. The needs to support mobile end devices like smartphones and tablets and to enable seamless workflows between different systems and platforms have introduced new levels of complexity. The A12 framework responds to these challenges by introducing flexible, light-weight UI Engines implemented in JavaScript – coupled with a coherent concept for device-independent visual design and modern look and feel.
Model business logic and user interfaces

Engines bring your models to life at runtime – using reusable widgets.

Adapt your models to changing business requirements

Designer and Business analysts

Tools

Models

Business Application
A12 project workflow and development

The A12 platform aims at speeding up the creation of web applications and reducing custom development efforts. How are these objectives reflected in the project workflow? And what happens if individual requirements enter the stage?

A12 projects typically start with business workshops where requirements are discussed and first specifications or user stories are sketched out. In contrast to classical software projects, these discussions have a stronger focus on considering a modeling perspective right from the start. Which business entities are involved and how can they be represented in the form of data models? How do different kinds of users interact with these data models? If users have to input data, which criteria determine whether the data is valid?

Using the A12 tools, first rough versions of data models, validation rules and online forms for interacting with these data models can be created. The result is a fully functional prototype. For further refinements, business experts and analysts iteratively adapt and extend the prototype depending on their users’ or customers’ requirements. This can be done completely without a development team. The application is updated by updating the models.

Dealing with individual requirements

Many business requirements can be implemented directly using the A12 modeling tools and existing building blocks. From time to time, however, new requirements may occur which cannot be covered with the current state of the platform. This is where individual software development comes into play.

The first step of dealing with new requirements is to determine whether they are project-specific or more general in nature. If the required features are very specific and probably not applicable in other A12 projects, they will be handled as custom developments – specially designed for the project in question. If the required features are potentially valuable in other A12 projects, they will be handled as A12 Platform Extensions. They are made available for general use.

Platform Extensions – How projects benefit from each other

A12 Platform Extensions ensure that projects don not have to reinvent the wheel. Instead, they draw upon more and more
thoroughly implemented and tested solutions for common requirements. This approach for transferring knowledge and sharing technical solutions is both beneficial for new projects and for existing projects based on A12. New projects have a larger A12 feature set at their disposal from the start. Existing projects may update or extend parts of their application with new A12 features without having to dive into time-consuming and costly custom developments.
A12 is all about models. But what exactly can you model as a business expert? What are the benefits of having different models for data and the user interface? And how does A12 differ from other modeling approaches like GUI builders?

The A12 platform enables experts and analysts to create and adapt business applications. It aims at building applications much faster than usual and allowing for business-driven adjustments on the fly. This is accomplished in large parts thanks to a model-based approach. The platform provides tools for creating models. And it provides technical environments which interpret these models and use them to control what the corresponding applications are about. The first step of gaining a deeper understanding of this approach is to take a closer look at the models.

“A guiding principle of the platform is the separation of different kinds of models: data models and UI models”, says Ansgar Weiss. Data models are created and edited with the A12 Data Modeler. They describe the underlying structures of the application. UI models are created and edited with the A12 UI Designer. They describe how the user interface is organized. “We decided to separate data and UI information from each other in order to introduce a higher level of flexibility and reusability”, explains Weiss. “You can create several different user interfaces for example which refer to one data model.”

The bedrock of business applications: data and validation criteria

The first step of creating applications with the A12 platform involves the creation of data models. If you want to build an application for stock management for example, amongst others you need a structural outline for articles. An article has a name, represented as a string. It has a serial number. It has a price. There have to be numbers which describe how many items of each article are in stock. Additionally, there are rules for the validity of data. The price can not be negative. The serial number has to satisfy a certain format. All these organizational structures and criteria which relate to the business perspective are defined in an A12 data model. In more technical terms, data models contain specifications of fields, data types, type
The A12 Data Modeler facilitates both the definition of data structures and validation rules. Data fields can be ordered in groups. For conveniently adding and modifying validation criteria, the editor features autocomplete and syntax highlighting for the A12 validation language. “In large applications there might be large sets of data definitions and validation rules. That is why we introduced so-called Includes: You can include data models in other data models”, explains Beate Still. If you have a basic data model for articles for example, you can reuse it in quite different contexts. “We regularly discuss additional concepts for modularization like inheritance”, adds Still, “but you have to be extremely careful in handling those mechanisms. Small changes can have large ramifications for the whole platform.”

**Content-related user interface design**

The A12 UI Designer pursues a rather abstract but very powerful way of modeling user interfaces. Instead of diving into design-specifics like colors, margins and spacings, the editor focuses on the organization of UI elements via models. Instead

**Benefits of a model-based approach:**

- Less custom development efforts: Create and modify models without programming.
- Applications can adapt quickly to changes from the business domain.
- Models survive technological changes. They can be reused in different setups.
- Model-based applications are well positioned against the backdrop of fast-changing technology landscapes.
of letting the user put input fields, radio buttons or text labels directly on screen in a WYSIWYG-style, it operates with hierarchical lists of model elements. This modeling philosophy enables business experts to focus on their domain and model complete user interfaces on their own. The graphical finish is completely decoupled and not specified in the A12 UI Designer. In order to provide an attractive and modern design out of the box, A12 makes use of Plasma Design – a coherent design concept for business applications developed by mgm.

Each A12 UI model refers to at least one A12 data model. This is a crucial aspect for understanding the design philosophy of the A12 UI Designer – and the main difference to classical GUI builders. There is no point in creating UI models without corresponding data models. You can think of UI models as a kind of wrapper for selected parts of your data models. They provide the frame for user interaction based on a selection of your data fields. Due to this connection, the data model and its accompanying validation rules are usually created first. As soon as this has been done, new UI models with a reference to this data model can be created. “Data and validation definitions really take the center stage in A12”, says Ansgar Weiss. “It takes a little getting used to, but once you get the idea of this unique approach, you will surely appreciate how fast you can build robust and productive software with it.”
Interface and interaction design usually take up a lot of time and effort in individual software projects. Plasma Design by mgm aims at speeding up the process. It is inspired by design languages like Google’s Material, but focusses specifically on business applications.

In every individual software project there comes a time when you get afflicted by a sense of déjà vu. The requirements look familiar. Tasks and discussions revolve around questions that have been answered in other projects. You get the feeling of having done it all before. As it turns out, this phenomenon of constantly reinventing the wheel does not only occur in software engineering, but also in the design and layout of user interfaces.

In which way does the application display data? Which colors and fonts should be used? What about the navigation layout, the design of buttons or boxes for content? In classical individual software projects, these aspects are negotiated with the customer. The design is more or less done from scratch again and again. This takes up half a year in the best case. “A core idea of A12 is to concentrate on the customer’s business requirements – not on design specifics”, explains Dietmar Schmidt, design expert at mgm. “Therefore we started to conceptualize a basic layout that should work in any project”. The idea of Plasma Design was born – a coherent design concept for business applications.

A neutral design for business software

What kind of layout and user interface does a business application really need? The mgm experts started to answer this question by reflecting about previous projects – mainly large-scale software projects in the insurance industry, the public sector and e-commerce. First of all, business applications are driven by function and by the necessity of performing certain tasks. It is more important to give users a reliable tool than to dazzle them with fancy images or flashy design effects. Functional considerations are more important than aesthetic considerations – which is not to say that a reduced, functional design cannot be appealing and elegant in a puristic way.

Second of all, most business applications share many structural elements. “Based on our experience from previous projects, we identified two major areas: forms and overviews”, says Schmidt.
In almost every business software you have to display aggregated data in a clear form, e.g. in tables, lists or charts. These overviews are generally used for providing summaries and supporting decision making or for selecting data sets for detailed editing. Detailed editing is usually done in forms – the dominant method of choice when it comes to data input from users. Since forms in business applications may get very large and complex with nested sections, defining a solid layout for any case imaginable is a challenging task.

**Multi-device support with desktop-first**

Another preliminary step of conceptualizing Plasma Design has been the analysis of existing frameworks and design languages. There is Bootstrap of Twitter, Metro of Microsoft and Material of Google. These frameworks offer elaborate and well thought-out concepts for graphical design and definitions about how control elements should behave in interactions. Especially Material Design has been received as a solid foundation by the mgm experts, except for a few aspects. “Everyone agrees that mobile-first characterizes the future of computing”, says Schmidt, “But it does not characterize the present when it comes to business applications. We still live in the world of the desktop.” Plasma Design therefore puts special emphasis on large screens while ensuring responsiveness and multi-device support.

On a conceptual level, Plasma Design is based on Material Design. But in order to focus on desktop-based web applications, a few minor tweaks have been applied – some fonts have been made smaller and some spacings have been narrowed etc. The UI/UX team of mgm has started to implement the concepts of Plasma successively. “We have refined and implemented the design and interaction concepts for forms and overviews already”, says Viet Nguyen, who manages the UI/UX work at mgm’s branch in Leipzig. “Technically, Plasma Design is provided as an npm-package for frontend developers. In putting the Plasma concepts into practice we rely on the Stylus CSS language and the BEM methodology”.

**Benefits for projects and future plans**

So, what is the main benefit for projects using Plasma Design? In a nutshell: speed. Projects can start immediately with a design
That is optimized for business applications. No individual design and interaction specifications are required. Prototypes which represent actual applications can be built from day one – which is one of the core ideas of the A12 platform. If customers like the design, they can keep it and use it in production. If they are not satisfied with its neutral, puristic aesthetics, an individual design can still be developed and used. This can be done by mgm or an agency for example. The main point is that a solid design for the business application in question is available from the start which drastically speeds up application development and prototyping.

While the current version of Plasma Design already covers the most important aspects of business applications, it is constantly developed further – both technically and conceptually. “In the future, we would like to introduce more capabilities for customization”, Schmidt explains, “You can tweak the design simply by adjusting sliders or by choosing between predefined layouts from a list”. Another field of work concerns the interplay of different content areas. How do different content areas relate to each other? What happens if one of these areas is

---

**Plasma Design Principles**

- **Function over Aesthetics**
  Business applications are tools. Clear workflows are more important than fancy layouts. Always use the maximum amount of screen space available.

- **Desktop over Mobile**
  Most business applications today are used on the Desktop. Web-based software has to be easy to use on large screens, but should also work on mobile devices.

- **Neutrality over Specificity**
  Plasma Design features a neutral design that can be used in any business context. Simple customizations in form of customer logos are possible.
resized? Or when data is changed in one content area and impacts other content areas? Can you specify rules or patterns for the interaction between these larger parts that constitute applications? These questions go beyond what is typically understood as UI/UX design, but they might lead to new modular approaches in application design and new insights about the building blocks that define today's frontends.
Tools
Modeling Tools for analysts

A12 provides a set of tools for business experts and analysts. Using these tools, they can create domain-specific models and build their own multi-lingual business applications. Programming skills are not required. The modeled business logic and structures of user interfaces can be easily reused.
**A12 Data Modeler**
- Create domain-specific data models
- Describe data structures of business entities like contracts or products
- Define fields and validation rules
- Predefined types plus custom type definitions

**A12 Validation Language**
- Reliable and versatile data validation
- Verify user input and assure data integrity
- Map business rules to validation criteria
- Autocomplete and syntax highlighting

**A12 UI Designer**
- Specify the structure and content of user interfaces
- Create complex online forms, including nested structures
- Create table- and list-like overviews
- Organize UI elements without having to dive into design-specifics
A12 Data Modeler

Create domain-specific data models
The A12 Data Modeler enables domain experts and analysts to create and modify domain-specific models for business applications. Programming skills are not required. Data models encapsulate central aspects of business logic. They describe the entities you have to deal with in business applications – for example contracts or products with all their relevant properties. Relying on data models brings about several benefits:

Less development efforts and adaptable applications
Domain experts can adapt applications by modifying models. They do not have to rely on developers for revising the implementation at every single change coming from the business domain. This is especially beneficial when it comes to confidential information inscribed in the models.

Better reusability and independence from technologies
Data models survive technological changes. They facilitate the migration to other systems and architectures. By encapsulating all domain-specific information in models, they can be reused in quite different setups. Model-based applications are well positioned against the backdrop of fast-changing technology landscapes.

Define fields and validation rules
A12 data models contain field definitions and accompanying validation rules. Validation rules range from simple restrictions – e.g. defining mandatory fields – to complex patterns and conditions across several fields. The A12 Data Modeler provides a set of predefined field types like “String”, “Number” and “Date”. They are complemented by optional custom type definitions. Fields can be ordered hierarchically in groups. Every time a new field is created, a dialogue window opens for comfortably defining its specifics. The resulting data model can be saved as an XML-file.
Features

- Create groups, fields, and relations
- Define rules for server- and client-side validation in one place
- Test validations
- Check for consistency
- Predefined types
- Type definitions
- Save A12 data models as XML-files
A12 UI Designer

Design User Interface (UI) models
The A12 UI Designer acts as a counterpart to the A12 Data Modeler. It is a tool for modeling the structures and the contents of user interfaces. UI models refer to A12 data models. They establish connections between the fields of data models and UI elements. Take an input field for example: An UI model describes its position in a form, its label and maybe additional user instructions in a text box. A data model specifies the underlying data type and the validation rules. There is a specialized UI model for every A12 Engine. The A12 Form Engine relies on an A12 form model. The A12 Overview Engine relies on an A12 overview model. Details of the visual representation of UI elements and the general look and feel are governed by mgm Plasma Design – a comprehensive design concept specialized for standardized business applications. UI models are saved as XML-files.

Master complex online forms
Forms lie at the center of many business applications. A12 Forms are comprised of common UI elements like input fields, buttons, labels, checkboxes etc. The A12 UI Designer offers powerful ways to organize these elements. You can create nested structures for example or hide whole sections, which are only displayed if certain conditions are met. Moreover, A12 Forms support repeated sections. This is quite useful if the number of certain sections varies.
Features

- Content-related design approach
- Organize UI elements without having to dive into design-specifics like colors, margins and spacing
- Multi-lingual labels
- Create complex forms, including nested structures and hidden sections, which are only displayed if conditions are met
- Save A12 UI models as XML-files
Reliable and versatile data validation

The A12 Validation Language is a powerful and versatile language for verifying user input. Based on business requirements it allows the definition of rules which can cover virtually all field-related validation tasks imaginable. The language ranks among the core features of mgm A12 – full-blown data validation is critical for preventing security risks and assuring data integrity in business applications. Using the A12 Validation Language, business analysts and domain experts are able to map their business requirements to validation rules. Error conditions may reference several different fields and groups.

Comfortable specification of rules

The A12 Validation Language contains many predefined predicates out of the box for handling typical validation tasks. It supports nested comparisons, arithmetic operations and provides special operators for handling dates for example. The language is directly supported in the A12 Data Modeler. It can be used in the “Edit Rule” dialogue for specifying error conditions. For conveniently adding and modifying these conditions, the editor features autocomplete and syntax highlighting. The A12 Validation Language has been developed by mgm in the context of transactional web applications. It has been used successfully for years in large productive software systems.
Features

- Powerful and versatile validation language
- Immediately usable in the A12 Data Modeler
- Convenient use with autocomplete and syntax highlighting
- Predefined predicates for fields, lists of fields and groups
- Arithmetic operations, comparisons, special operators for handling dates etc.
Building Blocks
Construction kit for digital business

A12 follows a modular approach when it comes to creating business applications. While business logic, validation rules and the structural layouts of user interfaces are encapsulated in models, the following building blocks are responsible for bringing the A12 applications and their models to life.
Widgets
Widgets are reusable, lower level UI components like grids, trees or datepickers that follow the Plasma Design conventions and UX concepts.

Engines
Engines act as self-contained runtime components that can execute models and data to create high level UI components like forms and overviews.

App-Templates
App-Templates speed up application development by containing engines, widgets, and interaction patterns.

Business Apps
Business Apps are flexible and highly adaptive web applications, driven by A12 models.
Widgets

A12 Widgets are reusable, lower level UI components like grids, trees or datepickers that follow the Plasma Design conventions and UX concepts. They represent the lowest level of A12 building blocks and are mainly targeted at developers.
The core technologies used for widgets are Typescript, which we use in A12 across the board for all code running on Javascript platforms, and React with its efficient rendering power. Widgets can be nested, i.e. a widget can have and use other widgets to provide sub-features.

The button widget for example uses the icon widget, the button group widget uses buttons. Another, more complex example: an application can use various input widgets to implement inline editing in a grid widget. To keep all the power of React, the composition of widgets is done on React level.
A12 Engines are self-contained runtime components – implemented in JavaScript – that can execute models and data to create high level UI components like forms and overviews.
Form Engine

The A12 Form Engine is designed to handle complex forms in client-side web applications. It can easily be integrated into websites as a runtime component.

Overview Engine

The A12 Overview Engine represents aggregated data in the form of lists and tables. It notifies the application when users trigger actions.
Realize complex forms in client-side web applications

The A12 Form Engine is designed to handle complex forms in client-side web applications. It is implemented in JavaScript and can easily be integrated into websites as a runtime component – for example in AngularJS applications. A12 Forms can be created and modified with the accompanying A12 tools (A12 Data Modeler and A12 UI Designer). UI elements like input fields, buttons and checkboxes are rendered in a standardized and modern business design.

Self-contained runtime component

The A12 Form Engine acts as a self-contained runtime component. It directly interprets A12 models: Details about the forms’ structure like the arrangement of input fields, labels, checkboxes or radio buttons are specified in A12 form models, a specific kind of A12 UI model. The underlying data structures and validation rules are defined in A12 data models. Both models can be created with the corresponding tools: the A12 Data Modeler and the A12 UI Designer. These tools and the server work with XML. The engine relies on JSON for data exchange. This makes it easy to bring document data into forms – even transformed and

Features

- Integrate complex forms in client-side web applications
- Light-weight implementation in JavaScript
- Easy integration as a self-contained runtime component
- Highly customizable
- Interprets A12 forms, supporting nested and repeated sections
- Data exchange via JSON objects
- Powerful rule-based data validation
- Modern and standardized UI design (mgm Plasma Design)
provisioned data generated from third-party logic for example – or output form data to the server side for further processing. The way in which the UI elements are rendered in HTML is guided by mgm Plasma Design – a device-independent design concept specialized on modern business applications. Due to its UI/UX implementation guidelines following the BEM methodology, CSS side-effects are ruled out.

**Typical use case: Web applications with complex or many forms**

The *A12 Form Engine* plays out its strength when it comes to web applications with many different or very large forms which are subject to frequent changes. You can get the most out of it if you use the related A12 tools and libraries. In principle the A12 Form Engine is a standalone component which can be used independently of other A12 components. Using the *A12 Form Engine* and the A12 tools is not recommended if the application exhibits only a few trivial forms with just a few fields.
Overview engine

Show document data in lists and tables

The A12 Overview Engine is a flexible component for representing aggregated data in the form of lists and tables. It notifies the application when users select rows or trigger actions like “delete” and “edit”. The engine is implemented in JavaScript and can easily be integrated into websites as a runtime component – for example in AngularJS applications.

Clearly presented, model-based overviews

Lists and tables play a key role in business applications. They offer a clear view on tasks or items in inventories for example or more generally on any selected aspects of filled-out forms. The A12 Overview Engine is a self-contained component for realizing those overviews in client-side web applications. The engine refers to two kinds of A12 models: An A12 overview model (a kind of A12 UI model) specifies the texts of labels and the organization of rows and columns. All underlying data structures are defined in A12 data models. Both models can be created with the corresponding tools: the A12 Data Modeller and the A12 UI Designer. These tools and the server work with XML. The engine relies on JSON for data exchange.

Typical use case: table- or list-like representations

The A12 Overview Engine is well-suited for web applications where data needs to be represented in a table- or list-like fashion. It can be used independently of other A12 components. Combining the engine with other A12 components like the A12 Form Engine is easy and quite beneficial. Imagine for example a simple application for managing contacts. Suppose you have a form for creating new contacts. Each created contact becomes a document composed of a number of filled-out fields like name, address, telephone number and so on. Using the A12 Overview Engine you can now reuse this data model and create a contact list which aggregates all the documents. Or you could create a phone list which only contains names and phone numbers.
## Features

- List- and table-like representations of document-data
- Supports selecting rows and triggering actions
- Easy integration in client-side web applications as a self-contained runtime component
- Light-weight implementation in JavaScript
- Interprets A12 overview models and A12 data models
- Data exchange via JSON objects
- Modern and standardized UI design
App templates

App templates contain engines, widgets, and interaction patterns that define the interplay between components.
Modeled business cases or use cases may act as building blocks themselves. All A12 projects benefit from each other, resulting in a growing set of digitization know-how.
The development process of business applications is currently facing significant technological changes. More and more functionality is moved to the client side based on JavaScript technologies. The trend is toward Single Page Web Applications. Equally important are well-wrought UI/UX concepts and the support of both desktop and mobile devices. The architecture of A12 addresses these challenges and provides the basis for developing modern web-based business applications.

State-of-the-art technology stack
The A12 architecture is built upon a carefully selected set of tried-and-trusted technologies. All engines are implemented in JavaScript and can easily be integrated into websites as runtime components - for example in AngularJS applications. Clients and server communicate via REST using HTTP with XML- or JSON-formatted content. Many functions can be used both on the server-side and the client-side. The A12 Validation library for example brings along a client-side JavaScript library and server-side Java library for rule-based validations.

How A12 supports the development of web-based business applications

- **A12 Plasma Design**
  Plasma is a design concept for business applications. It supports both desktop and mobile devices. Plasma provides visual components and concepts for user interaction.

- **A12 Widgets**
  Widgets are implementations of Plasma components – e.g. grid, button, datepicker, input field. They provide various levels of API for integration in React and Angular.

- **A12 Engines**
  Engines are based on Plasma and A12 Widgets. They enable business experts to define parts of the application in models using a set of convenient tools.

- **A12 CAT**
  The Client Application Template (CAT) provides a project setup and integrated tooling supporting development, build and release processes for JavaScript environments.
The A12 architecture puts a strong emphasis on simplifying client-side application development. It features a pre-configured build setup, engines for working with models and widgets for reusable UI components.

Standards where possible – individual solutions where necessary

All parts of A12 – CAT, Plasma, Widgets, and Engines – are designed for **extensibility** and **customizability**. A12 aims at providing stable solutions for recurring problems while allowing for individual application-specific additions where necessary:

- Application-specific UI parts can use Plasma directly
- Application-specific components can coexist and interact with Widgets and Engines on the same screen

Plasma, Widgets and Engines provide **structures and conventions**, which can be used and applied to project-specific components and code.
The Client Application Template (CAT) and the Client Application Framework (CAF) significantly speed up the development of client applications by providing a consistent frame and setup.

Professional development of sustainable business applications in the emerging JavaScript ecosystem is still in its early stages. Due to the high popularity and prevalent hype culture many diverse solutions and short-lived technologies are flooding the market. Every web client project faces series challenges and has to answer a bunch of organizational questions. What is the language of choice? What about the platform? Browser, Node.js or Cordova? The list goes on with frameworks like Angular JS and React, the build system (Gulp, Grunt, Maven, Make etc.) and different types of project structures. CAT aims at providing a standardized and sustainable solution for developing web-based clients.

Streamlined build processes

The build process is typically a core challenge in web client projects since several languages like Typescript, JavaScript, Jade and Stylus are in use. Compared to stacks in the Java or C# world no standards have gained widespread acceptance yet. This results in heterogenous build processes across projects and many different solutions for similar problems. The Client Application Template provides automated build processes. We constantly analyze the JavaScript Ecosystem with regards to the requirements of business applications in order to create a solid template.
Client Application Template (CAT)
CAT is a preconfigured build setup. It provides support for the following tasks in web application development:

- Setup a new project / update an existing project
- Automated build, packaging (npm-based), advanced componentization and bundling
- Typesafe development
- HTML and CSS templating
- Rapid development (hot deployment, browser syncing)
- Code documentation
- Complete stack for automated headless and browser tests execution and reporting

Client Application Framework (CAF)
The Client Application Framework is a collection of node modules for using the A12-Engines in AngularJS-based SinglePageApplications. It is intended for Typescript-based JavaScript projects which work with Stylus-based CSS transpiling. CAF is used to configure, customize and orchestrate the A12 engines. It brings along advanced navigation features with support for deep state retaining.
A12 features server-side applications which communicate with the A12 tools and A12 client applications. The server application is based on Java and a set of reliable technologies like Apache Solr – one of the world’s most popular enterprise search platforms. It supports XML and JSON as input and output format for models and documents. The core functionality is accessible via a REST API. The REST API uses the HATEOAS HAL approach to enable clients discovering it.