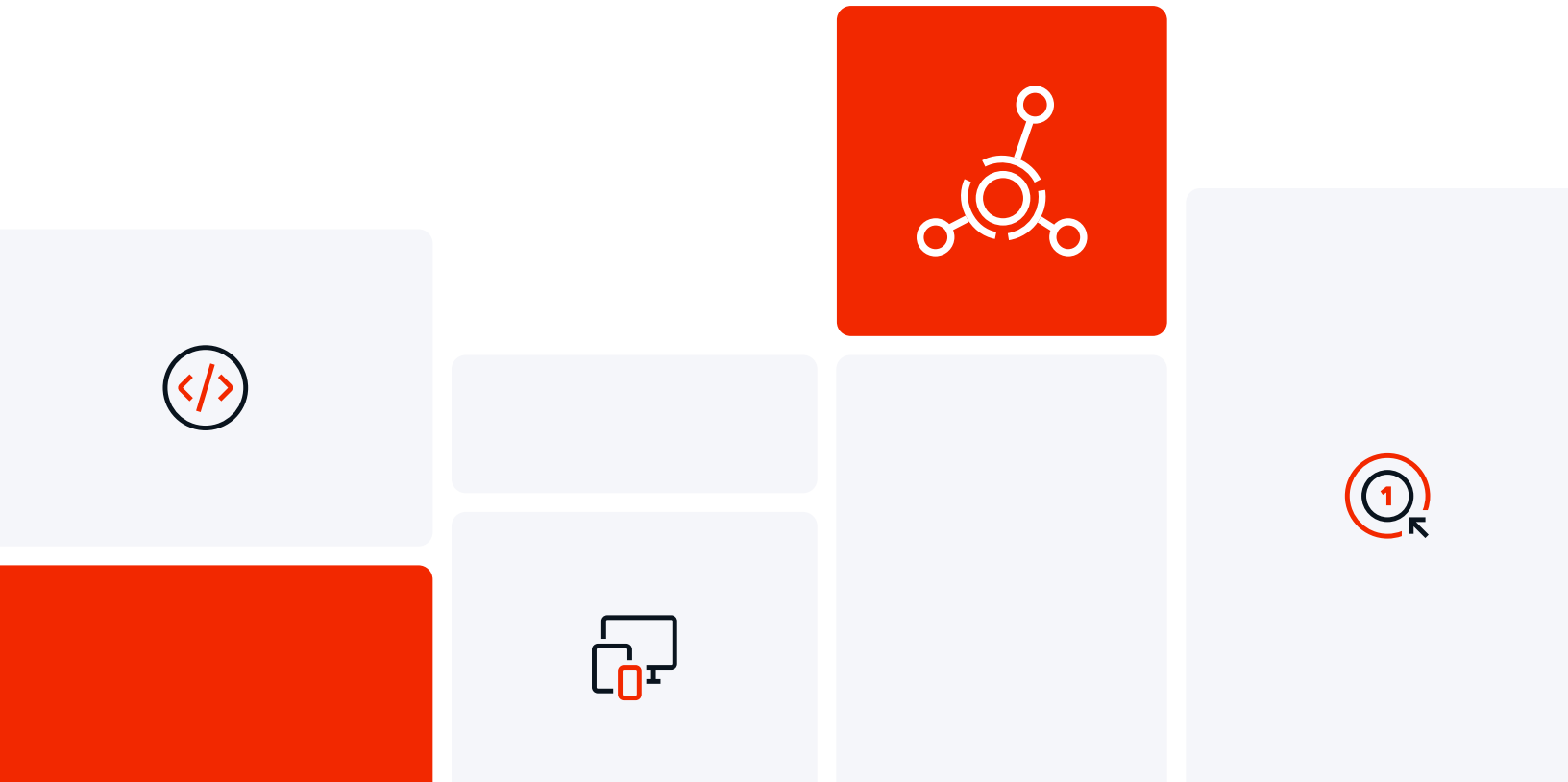
outsystems

OutSystems Extensibility

TECHNICAL WHITE PAPER



Contents

Introduction

Creating an enterprise-grade application that both serves customer needs and fulfills business goals is a painstaking process, and it involves multiple integrations with your back-end and your data. Although there are no shortcuts to success, a modern application platform can make these integrations easier.

OutSystems is a modern application platform that lowers the initial cost of setup and training, speeds up design and development, accelerates deployment, and simplifies maintenance. Using visual tools to build a rich user interface and model-driven logic with pre-built components opens the door of your development team to talented people who don't have formal programming skills.

Unlike other development tools that offer shortcuts, with OutSystems, you do not have to abandon legacy systems or lose access to valuable databases. You can integrate with everything your enterprise has developed over the years, and you can keep up with anything that comes up in the future.

Because you can't always drag-and-drop your way to a solution, sometimes you need to finesse it with classic code. OutSystems can be extended, if needed, by a little customized programming—which can then be integrated into the development environment and reused as just another visual tool.

Outsystems is just this type of platform: open by design and extensible for every need in all application layers—front-end, back-end, database, and integration. This paper explains the extensibility of the OutSystems platform in detail.

Build anything

Those responsible for planning, defining, and maintaining architectures for an organization's IT networks and services must be sure, when considering the adoption of a new development model, that it will allow easy and efficient creation of enterprise-grade applications with rich and complex mobile and web user interfaces. It must also provide seamless integration with the myriad of existing code, database structures, and systems of records, such as ERP, CRM, and HR, that the enterprise has developed over the years—as well as keep pace with future developments.

Fourth-generation language technology platforms boast a low code replacement for software development with drag-and-drop interface design and push-button database integration using pre-built components. However, **4GL** also comes with pre-built drawbacks, not the least of which include sluggish operational performance, cybersecurity risks, and inaccessible support for third-party components.

OutSystems is based on a visual domain specific language that is built on top of standard third generation language (3GL) programming elements.

It's suited to develop anything from simple departmental applications to more complex enterprise applications, containing all of the necessary building blocks to implement sophisticated use cases from rich engaging UI experiences (mobile or web), headless backend services, workflows, or batch processing mechanisms. All this without the need to write a single line of code, only using drag and drop visual elements to build UI, business processes workflows, database modeling, or intricate logic to support your use case.

However, OutSystems doesn't lock developers into a limited number of blocks, modules, and pre-built components. If needed, developers can use their coding skills to extend built-in capabilities to fulfill specific use cases.

OutSystems is open by design. All application layers—front-end, back-end, database, and integration—can be extended with customized code. In fact, **OutSystems Forge** contains a repository of open source libraries, plugins, and pre-built extensions that can be freely used as-is, or modified to suit particular needs, and then added to the OutSystems tool pallet just like any other native OutSystems element.

Extensions

OutSystems **Integration Studio** is a design environment where developers with programming skills can create **extensions** to bridge custom C# and .NET to OutSystems, expanding and enhancing its existing functionality and data model.

An extension, as the term is used in Outsystems, is a specific set of actions, structures, entities, and resources that developers create in order to:

- Use custom-built .NET components to extend module functionality
- Integrate with external systems to extend module functionality
- Integrate with an external relational database to extend the module data model

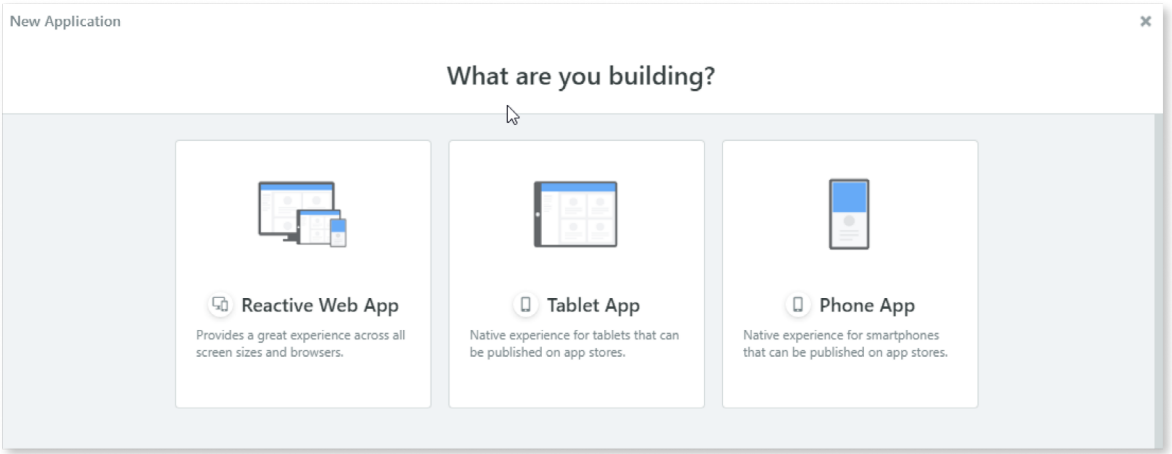
As an example, Integration Studio provides automatic wizards that **reverse engineer external databases** using code introspection to **translate .NET assemblies** into extension actions, entities, and structures.

And by using the **1-Click Publish** feature, a newly created extension, along with all of its dependencies, can be immediately deployed to any platform server. Once published, the extension can be used by any module that has access to it and can be published in multiple installations of the platform.

Front-end design

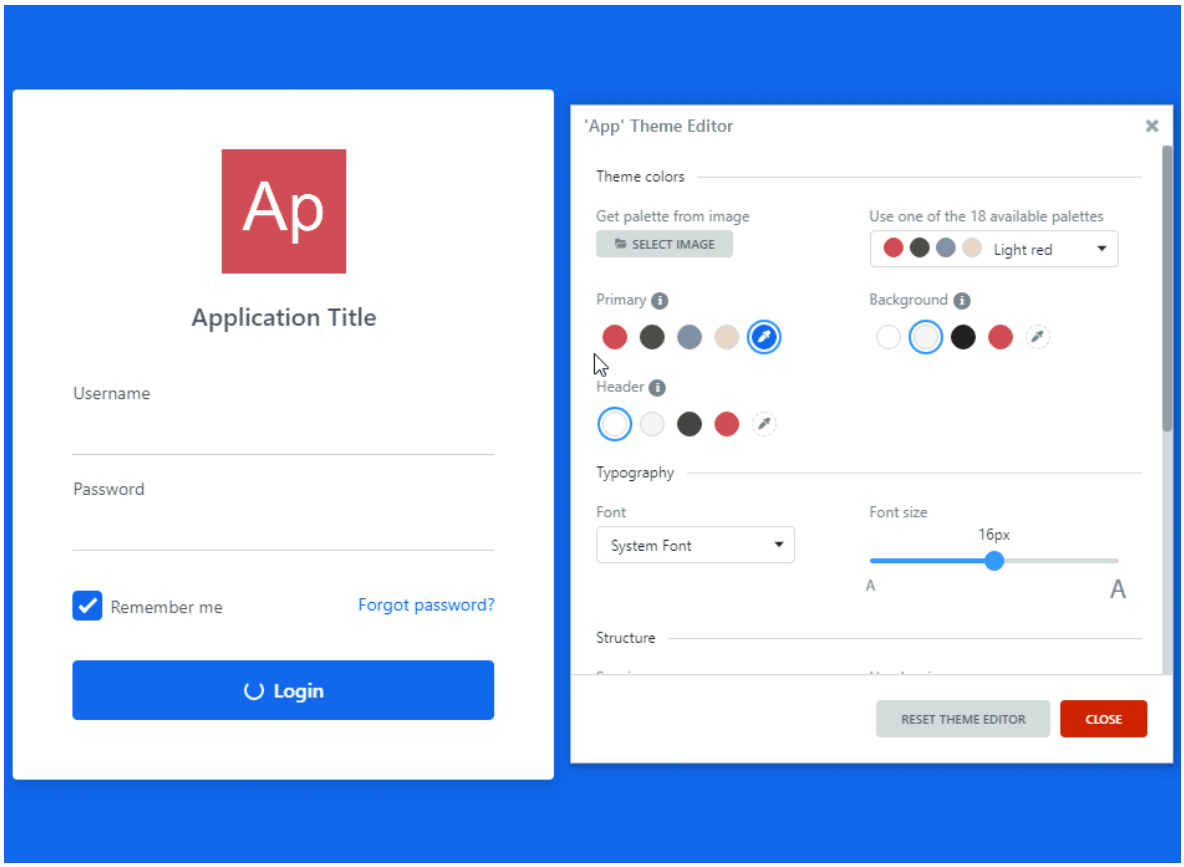
OutSystems UI enables the creation of complex mobile and reactive web user interfaces by providing a large collection of screen templates, widgets, and UI patterns that your developers drag and drop in a visual editor to create beautiful applications that work across devices without the need to write a single line of code.

A typical workflow begins by selecting an **application template** that contains common elements, such as screen layouts, CSS, and basic logic, which developers use to quickly build reactive web apps, phone apps, or tablet apps that conform to company style guides and business rules.

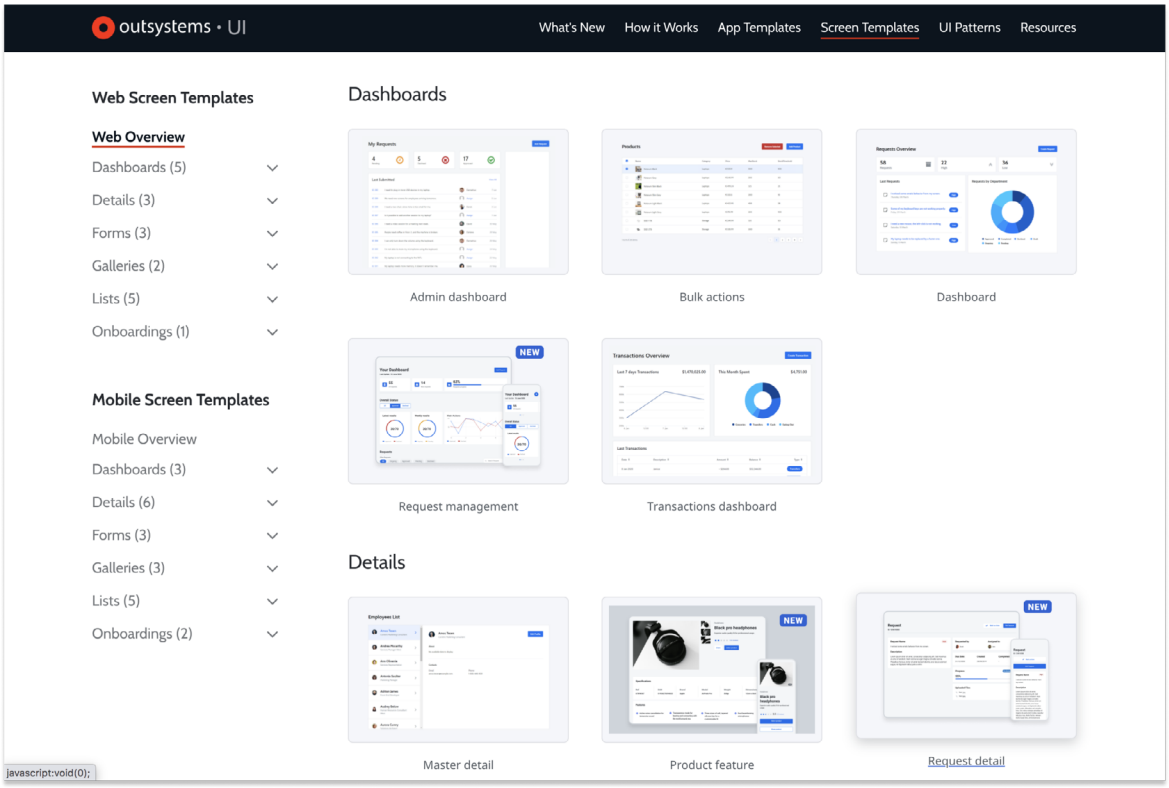


In addition to the built-in application templates, developers can choose from hundreds of **Forge application templates**, provided by the OutSystems community, to install and modify to create a new app, just like other application templates.

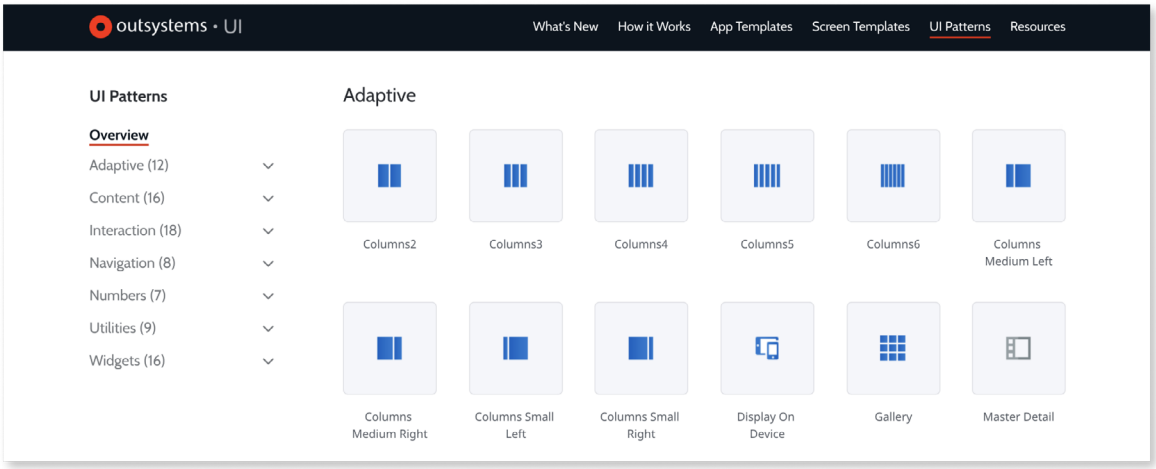
The look and feel of the app is set in the OutSystems theme, which is part of the **OutSystems UI** framework. Using the **Theme Editor**, the overall style of the app can be customized in Service Studio, including default colors of screens and UI elements, default font and size, spacing and borders.



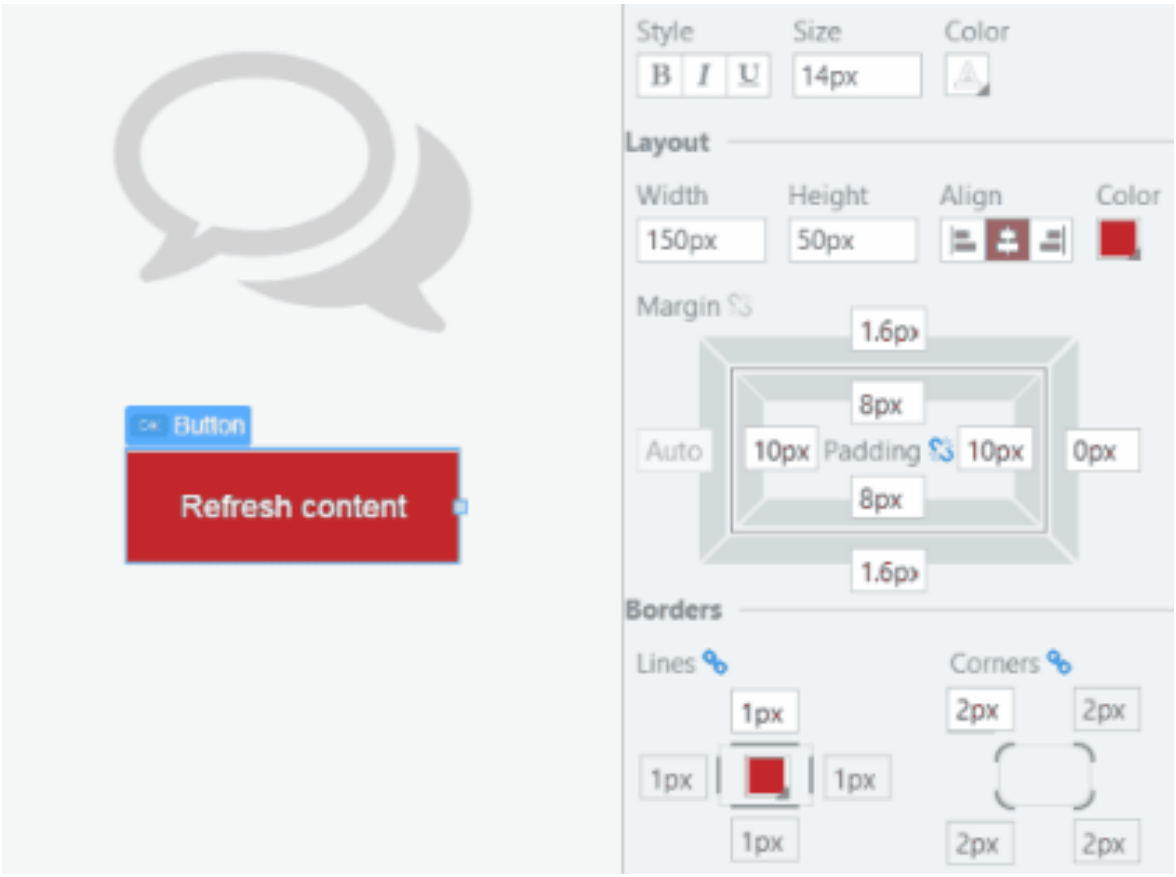
Screen templates for both web, mobile, and reactive apps provide theme-consistent UI and for such functions as dashboards, details, forms, galleries, lists, and onboarding. Each screen is populated with widgets and containers that can be modified and rearranged as required, including sample data that developers can use for testing on a computer or mobile phone.



Sometimes none of the built-in screen templates answer the needs of a particular function or activity. In this case, a new **custom screen template** can be built from scratch by dragging and dropping **UI patterns** to set the page layout, types of content and interaction, navigation and numbers, utilities and widgets—all from an extensive selection in the Service Studio toolbox. Once the new screen template has been published, it is available for use in any app in your environment.



The look and feel of widgets can be customized using the Styles Editor panel for editing basic visual properties, such as font color, margins, or border thickness. The changes can be made locally to affect only the selected widget, or saved globally for all widgets of that type used in the app.



Once developers have customized one of the OutSystems application templates, created screens, built logic, and published the app, they can save it as a **custom application template** that can be used as a starting point to develop additional apps, to define its look and feel, and put in place common functionality and manage dependencies.

Front-end extensibility

The OutSystems UI front-end development process, as described above, shields developers from the complexities of coding and allows for extensive customization of built-in components of everything from the default look and feel of screen templates and UI patterns to the functionality of widgets—all within a WYSIWYG development environment.

There is no way, of course, to anticipate every possible design choice or function that might be required for a complex enterprise app. OutSystems, instead, gives developers an easy way to extend built-in features and fill in the gaps by writing code.

HTML extensibility

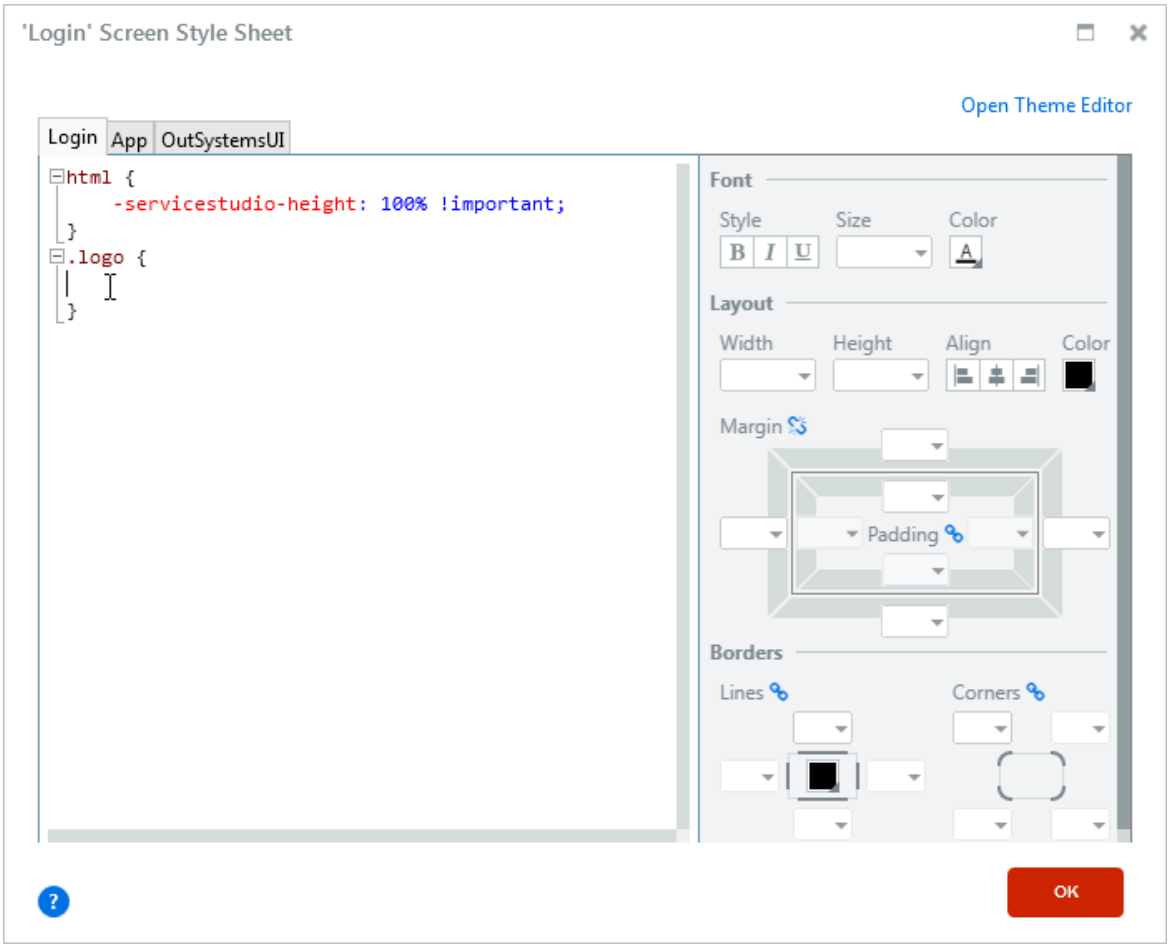
All can be customized by setting extended properties which are directly translated to HTML attributes. This includes HTML tags that are not currently supported in the base HTML definition, such as custom “data-” attributes or accessibility features. An HTML widget for reactive and mobile applications can include any HTML code for specialized functions, such as a CMS that loads dynamic, database-stored HTML content in a page.

CSS extensibility

Developers use **OutSystems UI** to tweak the styles of individual elements or make global modifications to the theme of the chosen application template.

In some cases, however, low-code customizations may not suffice to meet specifications set in an app mockup. For example, the enterprise brandbook may require a proprietary font set, or the corporate color scheme in the style guide may not be found among the 18 pallet options available in the Theme Editor.

For these situations, new styles can be created, or existing classes modified, in the **CSS editor**, which overrides the properties in the base CSS, for themes, screens, emails, blocks and widgets. The OutSystems UI makes extensive use of CSS variables for **color**, **typography**, **borders**, **spacing**, and **shadows**, and these can be modified within the CSS editor.



In some cases developers may decide to use an external CSS that is compliant with OutSystems UI by using the @import rule in the OutSystems UI CSS.

JavaScript extensibility

All modern web design is built on a foundation of JavaScript, the engine that makes user interaction possible—and OutSystems UI is no exception. In most cases developers have no need to know how JavaScript functions within a widget or Web Block; it just works.

For customized work flows many additional JavaScript-based widgets have been packaged into easy-to-reuse Web Blocks by OutSystems, and the Outsystems community, and published to **OutSystems Forge**. Examples include **Drag and Drop lists**, **Touch Drag and Drop**, **intro.js**, and the well known **Google Maps**.

Developers are also free to create extensions by writing JavaScript to customize how users interact with their applications, create client side custom validations and dynamic behaviors, or even develop custom, very specific, AJAX interactions. OutSystems includes jQuery by default in all applications, but developers can wrap their own JavaScript frameworks, including prototype, jQuery, jQueryUI, dojo, React, Angular, and Vue, among others, in a Web Block. Once published, it becomes available in the visual interface and can be used by the entire development team, throughout all applications, just as they would in any HTML page design.

By **importing a library into an application** and adding it to the web screen in **Service Studio**, all of the library's functionality becomes available in every OutSystems application, making a custom JavaScript as easy to use and reuse as any other component in the OutSystems model-driven development.

Extending the back-end

OutSystems has out-of-the-box capabilities that allow seamless integration via SOAP web services, REST APIs, and SAP systems. An integrated visual IDE in OutSystems allows rapid and efficient creation of extensions to integrate with core or legacy systems. When using the OutSystems environment, developers can click a link to open a source code editor, and, after editing the code, the developer can deploy it in the integration IDE.

OutSystems simplifies the task of consuming and exposing functionality as standard SOAP web services, REST APIs, or SAP services without any hand-coding. More complex enterprise integration scenarios that require MQSeries, TIBCO, BizTalk, and other middleware ESBs, can be accomplished using web services.

Once OutSystems has generated the methods and data structures to integrate with the external system, developers can use them visually in the application logic of Service Studio. From a developer's perspective, there is no difference between invoking an OutSystems method, a SOAP web service, a REST API, or an SAP BAPI.

Integrating to existing systems using SOAP web services

OutSystems provides out of the box support for consuming SOAP web services with WSDL 1.1 and SOAP 1.1/1.2.

Developers provide the location of the Web Services Description Language (WSDL). The OutSystems IDE inspects the WSDL and generates everything needed to invoke the web service methods.

When you consume a SOAP Web Service, OutSystems does the following:

- Creates the SOAP Web Service
- Creates the exposed Web Service methods with the corresponding input and output parameters
- Creates the structures and static entities to hold the complex types defined in the WSDL
- Maps the XML data types into OutSystems data types

OutSystems translates the methods exposed by a SOAP Web Service into OutSystems actions, with the same semantics as any action created in Service Studio. From a developer's perspective, there is no difference between invoking a system method or a method exposed by an external service.

Authentication options are provided without the need of complex extensibility such as basic authentication (a static username and password) or dynamic login for credentials specific to each method that can be specified at runtime.

When developers want access to Web Services Enhancements (WSE) or need to customize the behavior of consumed SOAP Web Services, they can create an extension by using the **SOAP Extensibility API**.

You can see a number of SOAP extensibility examples, such as authentication using a client certificate or using a different character encoding, on **this page**.

OutSystems has created the **SOAP Extensibility Samples extension**, available for download from OutSystems Forge, that developers can use as is, or adapt as required, to implement common extensibility use cases using the SOAP Extensibility API.

Extensions may be used for any other use case that is not either provided out of the box or by OutSystems APIs.

Integrating to existing systems using REST APIs

You can integrate your OutSystems applications with REST APIs provided by other systems, or even by other OutSystems applications. Use this functionality either to obtain data from those systems or to request them to perform some action.

When importing the REST API, Service Studio does the following:

- Creates the REST API
- Creates REST API Methods with the corresponding input and output parameters
- Creates the Structures to hold the corresponding input and output parameters under a new tree element with the REST API name
- Maps the REST data types into OutSystems data types

OutSystems translates the methods exposed by a REST API into OutSystems actions, with the same semantics as any action created in Service Studio. From a developer's perspective, there is no difference between invoking a system method or a method exposed by an external service.

When consuming a REST API, you can add logic to customize the information that is sent in the requests or received in the responses. **Simple customizations** are readily available in Service Studio to modify both the information of the original request (such as the URL, the request text or the headers) or the information of the original response (such as the status code or the response text).

When developers need to further customize the REST API they can create an extension using the **REST Extensibility API** and their own .NET code. These

advanced customizations then become available to be reused in Service Studio. These advanced customizations allow, for example, using a specific algorithm to encode part of the request content, adding support for client-certificate authentication, or handling a complex response.

There are hundreds of extensions that use the REST Extensibility API that you can browse in **OutSystems Forge**.

Integrating OutSystems applications with SAP

With OutSystems, developers can create mobile and web applications that integrate with SAP by browsing available SAP functions. SAP OData services can be imported from the **SAP Business Hub** and Service Catalog. Finding the BAPI or ZBAPI for a specific function is simple, even with limited or no knowledge of SAP. Integration can be through SAP BAPIs or SAP OData services; both are available for SAP ECC and SAP S/4HANA.

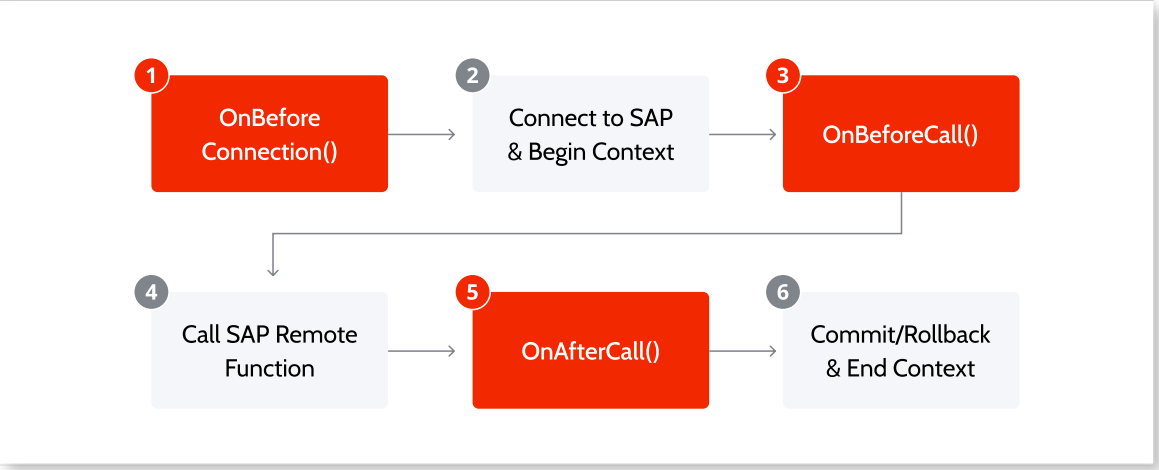
Once a connection has been made and remote functions have been imported OutSystems does the following:

- Creates the SAP connection you have configured
- Creates the actions for the remote functions you have selected
- Creates the structures to hold the parameters

These newly created actions can be used in your application the same way you use any other server actions.

The flow when calling a SAP remote function allows you to:

- Implement different SAP authentication methods (OnBeforeConnection)
- Customize the values to be sent to SAP (OnBeforeCall)
- Customize the values returned by SAP (OnAfterCall)



When integrating with SAP, OutSystems uses, by default, the credentials provided when the connection to the SAP system was created. You can also use different credentials by configuring a different connection by environment or using a dynamic login.

If developers need to customize a SAP connection, calls to SAP remote functions, or implement different SAP authentication methods, they can create an extension using the **SAP Extensibility API** and their own .NET code.

OutSystems has created the **SAP Utilities** extension, with ready-to-use actions to perform customization of calls to SAP remote functions. Developers can use it as is to add advanced functionality to SAP remote functions. or adapt it, as required, to implement common extensibility use cases using the SAP Extensibility API.

There are numerous extensions that use the SAP Extensibility API that you can browse in OutSystems Forge.

Integrating external data

OutSystems connects, out-of-the-box, to SQL Server, SQL Azure, Oracle, MySQL, and DB2 iSeries so developers can combine local and external data sources in minutes on complex integration projects.

The OutSystems visual editor allows development teams to **query and aggregate data visually** so developers with any skill set can work with the complex data needed for any application.

Using SQL

The OutSystems language provides actions for extracting, loading, transforming and updating data, but developers are not limited to manipulating data visually using aggregates. More experienced developers can also use an **SQL Node** to build complex queries or to run stored procedures.

OutSystems continually tracks all of the dependencies to the underlying database. If something changes, your SQL code is automatically refactored, or you're warned if it is no longer compatible with the changes. This extensibility feature accelerates the change process, as well as makes the application itself resilient to change.

Integration with external databases

If an external database is natively supported by OutSystems (SQL Server, SQL Azure, Oracle, MySQL, and DB2 iSeries), a developer can **create a connector** to integrate with it. Connectors enable teams to manipulate data in the visual editor.

An on-premises database that isn't natively supported can be accessed, using **standard code**, with ODBC/JDBC drivers, or with native APIs. Developers can

access big data repositories, such as CouchDB, MongoDB, or Hadoop, using their REST APIs.

By developing a custom connector, designers can extend the OutSystems platform to add native support for databases. Many such connectors have already been built by OutSystems users and shared, for your use, in OutSystems Forge.

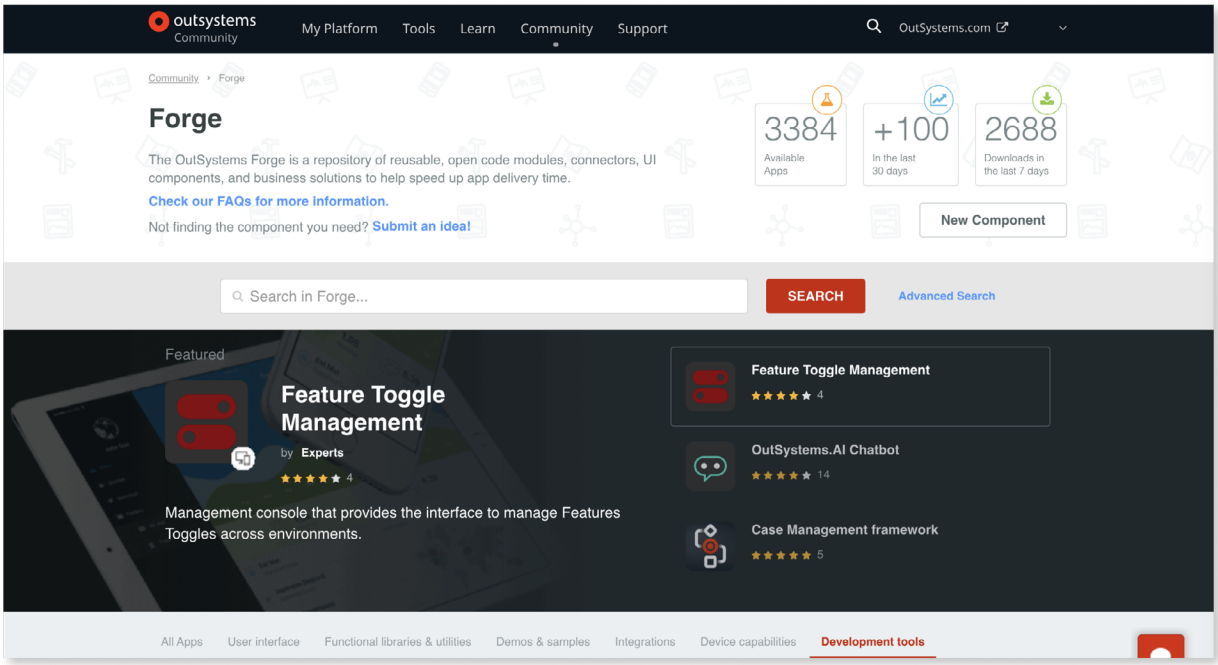
Importing and exporting XML and CSV content

Developers can easily import and export data from Excel or JSON directly from the toolbox, and can use these integration components in OutSystems Forge to import and export content in XML or CSV format.

If developers need tighter control of an Excel export, they can use **Advanced Microsoft Excel**, available in the Forge. This extension supports data formatting, the creation of Excel charts, and ability to fill in specific cells by providing row and column numbers.

Using OutSystems Forge to extend applications

The **OutSystems Forge** is a repository of reusable, open code connectors, widgets, libraries, themes and templates, mobile plugins, applications, reporting, and developer tools that help speed up app delivery time that can be installed and reused as if they belong to OutSystems itself.



Among the more that 3000 currently available extensions developers can find the following:

- **Connectors** to integrate with external systems. They allow information to be fetched from or stored in external systems and can provide additional functionality like caching. Examples of connectors are the **Salesforce Connector** and the **Jira Connector**.
- **Widgets** that provide functionality beyond those available in Service Studio, such as **Google Maps** and **Signature Pad**.

- **Libraries** to extend the actions and built-in functions of the OutSystems platform, such as **JavaScript Utils** and **CryptoAPI**.
- **Themes** that include page layouts and all the visual assets necessary for your application. Examples of themes include the **OutSystems UI** Web and the **Lisbon Template**.
- **Mobile plugins** to extend the capabilities of mobile applications by giving access to the device's sensors and capabilities. Examples of include the **Camera Plugin** and the **Key Store Plugin**.
- **Applications** of fully working apps that you can install and start using immediately. Examples of applications include **Field Services Mobile** and **Floor Plan**.
- **Reporting** includes components that help build reports for end users, such as **Pivot Table** and **HTML2PdfConverter**.
- **Developer Tools** to help ensure that developers are following best practices in Outsystems, such as **Discovery** and **DBCleaner**.

In addition, the Forge includes **How-tos** that give detailed examples for using a particular capability, explaining a component's functions, or the steps required to perform a specific task in OutSystems. Examples include **How To - Build an App with Multilingual Support** and **How to - User Sign Up**.

If what you need isn't available on OutSystems Forge, you can use **Integration Studio** to create your own extension, or modify an existing one, with custom C# code.

OutSystems native mobile apps are built on top of the Apache Cordova framework, which enables using any native device API through the use of plugins. Developers can **create their own plugins** by selecting one of the available **Cordova plugins** and wrapping the code in an OutSystems API. From that moment onward the new native functionality is available in all the OutSystems.

Conclusion

OutSystems is based on a visual domain-specific language, built on top of standard 3GL programming, that contains all of the necessary building blocks to implement sophisticated use cases without the need to write a single line of code.

However, OutSystems developers aren't locked into a limited number of blocks, modules, and pre-built components. All application layers—front-end, back-end, database, and integration—can be extended. Thousands of pre-built extensions, available for free download from the OutSystems Forge, can be used as-is, or modified to suit particular needs. And, for special cases, a developer with classic coding skills can create a custom extension and add it to the OutSystems tool pallet so it, too, can be used and reused just like any other native OutSystems element.