Creatio Academy

App development in Creatio

Creating applications on Creatio platform

Version 8.0

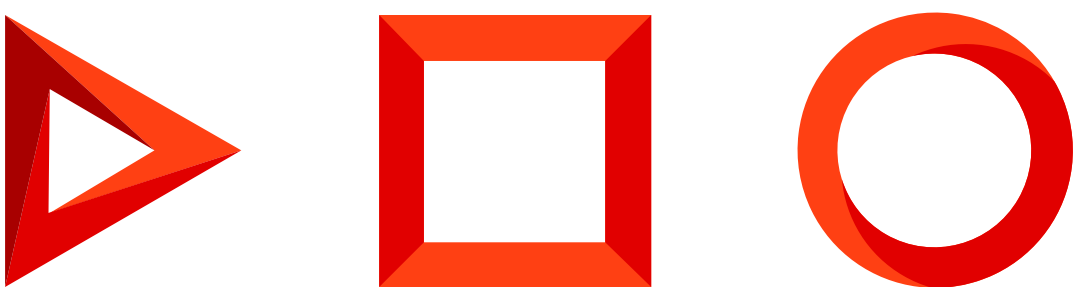


Table of Contents

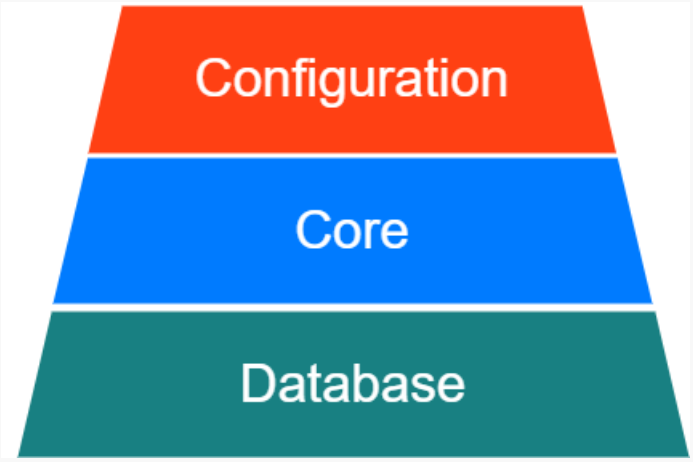
Creating applications on Creatio platform

Beginner

Creatio is a low-code platform designed to accelerate application development, implementation, and scaling. The platform is built with customization, flexibility, and scalability in mind. This makes application development possible for developers of varying skill levels – from a business analyst to a full-stack developer. Application development in Creatio allows for different levels of customization depending on the complexity and type of business goals.

Creatio customization levels

Creatio architecture comprises several logical levels of interaction:



Note that the Creatio core level is an unmodifiable component. Development in Creatio is done on the configuration and database levels.

Database

Database is the physical data storage level. The database stores client data, application settings, and access permissions.

Creatio tools let users work with data in the UI. As such, you do not need to work with database objects directly.

For certain tasks, the database level development is the most logical solution and the fastest method. You can implement custom business logic on the database level using views and stored procedures. Afterwards, you can call the business logic from custom configuration elements.

Core

The **core** is an unmodifiable part of Creatio. It is a set of libraries implementing the base Creatio functionality.

The **back-end libraries** are written in C# with the .NET Framework platform classes. Developers can create back-end class instances and use the the back-end libraries, but they cannot make any changes to these classes

and libraries.

The main **back-end core components**:

- ORM data model and its operation methods. In most cases, we recommend using the object model, though the back-end core components also allow for direct access to the database.
- Packages and replacement mechanism.
- Creatio web services.
- The main designer and Creatio section functionality.
- Third-party service integration libraries.
- Business process service (`ProcessEngineService.svc`). This Creatio element executes algorithms depicted as flowcharts.

The **front-end core classes** are written in JavaScript on top of different frameworks. Developers can use these classes to create UI and implement other browser-side business goals. The main **purpose** of the front-end core components is to manage the operation of client modules.

The main **front-end core components**:

- External libraries of client frameworks.
- Sandbox – a special client core component that manages the message exchange between client modules.
- Base modules – JavaScript files that implement the functionality of the primary Creatio objects.

Configuration

The **configuration** is functionality available for Creatio users of a specific workspace, namely:

- Server logic.
- Classes generated by Creatio settings automatically.
- Client logic (pages, buttons, actions, reports, business processes, and other configuration elements).

The configuration is an easily modifiable part of Creatio. A specific configuration constitutes the following element types:

- **Objects** – data-storing entities that combine a table in the database and a class on the server's end.
- **Business processes** – configurable elements that depict a user action flowchart.
- **Client modules**.
- **Server modules**.

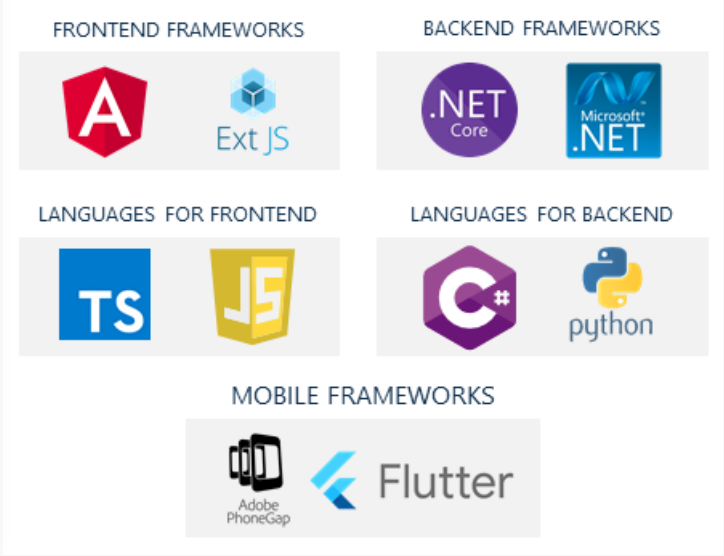
Application development tools

Creatio provides a wide variety of tools to create new and modify existing applications.

Open source technologies

Support of standard programming languages and frameworks streamlines the development and troubleshooting, as well as makes it easier to find qualified specialists with the necessary technology stack.

Creatio supports the following **technologies**:



The **built-in IDE** lets you implement complex business logic, integrations, and customizations, accelerating many common configuration procedures. Use C# and JavaScript to solve the following **problems**:

- Expanding and modifying Creatio functions.
- Organizing the interaction with version control systems.
- Migrating changes between the development, testing, and production environments.

Creatio supports **third-party IDEs** (e.g., Microsoft Visual Studio, MS Code, Rider) that let developers work with projects in a local file system. The familiarity of the IDE, as well as the variety of plug-ins, extensions, and version control system integrations, etc., streamline the development. Developers can save time spent on learning new tools and concentrate on coding.

Low-code/no-code development

The low-code/no-code development **tools** are a set of drag-and-drop UI editors. They let you solve the following **problems**:

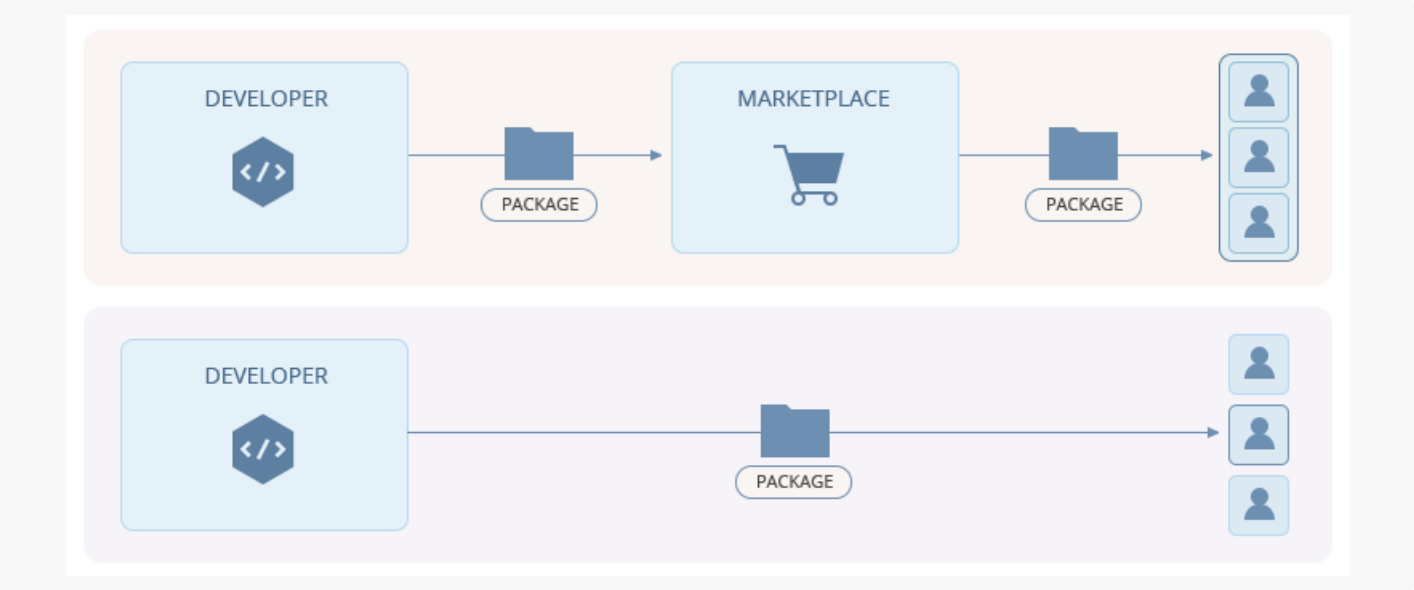
- Business process and dynamic case automation.
- Data structure modeling.
- UI modeling for web and mobile apps.
- Dashboard creation and report building.
- Integration setup.
- Predictive model building, etc.

The low-code tools cover most use cases. Minimal coding, as well as easily accessible built-in tools, let business experts, power users, analysts, and citizen developers. Learn more about no-code tools in the No-code article.

Package mechanism

Regardless of the tools used to customize the platform, Creatio bundles all changes into one or more packages – key Creatio architecture components. The package architecture lets you create separate modules from autonomous code blocks, as well as control the package hierarchy and versioning. That way you can expand the configuration, as well as migrate changes between the development, testing, and production environments, faster. The Marketplace solutions are based on the same mechanism.

Any Creatio product is a set of packages installed on top of the Creatio core. A package contains object schemas, the source code, business processes, reports, etc. It also may contain third-party assemblies, SQL scripts, system settings, and custom data.



Creatio extension model is based on the **open-closed principle**, where major application logic is closed for direct manipulations but open for extensions with custom packages. A package from another publisher (an integrator partner, a Marketplace developer, or a customer) can expand any package. This allows the platform to combine out-of-the box products, marketplace solutions, and client customizations effectively in almost any variation.

The **package architecture** is the main way to deliver and deploy custom applications, extensions, and templates, seamlessly integrated with Creatio Marketplace. Creatio Marketplace is an ecosystem for developing, distributing, and acquiring customizations – from custom apps and templates to updates and changes to industry applications.

Packages do not affect the core platform logic, allowing for smooth delivery of updates to Creatio Core, as well as parallel deployment and updates to generic functions.

DBMS–independent architecture and Creatio's ORM

Creatio is a **DBMS-independent platform** based on its own ORM. This allows the developers to build and deliver custom apps to various Oracle, PostgreSQL, or MS SQL Server configurations easily without any changes to the codebase.

Process-based business logic

This is an environment where full-stack developers and business analysts interact. Users can create their own

business logic by simply designing the needed processes in a drag-and-drop business process editor. Learn more about working with business processes in the BPM tools article block.

Integrations

Creatio provides all necessary tools for third-party system and app integration, including the support of the REST API, OData protocol, SOAP services, OAuth and LDAP authentication. You can develop integrations as parts of either Creatio or a third-party app. Complex tools ensure data security both during the integration for identification and access control, and during the user structure management.