Creatio Academy

Recommendations

Recommendations on app creation

Version 8.0

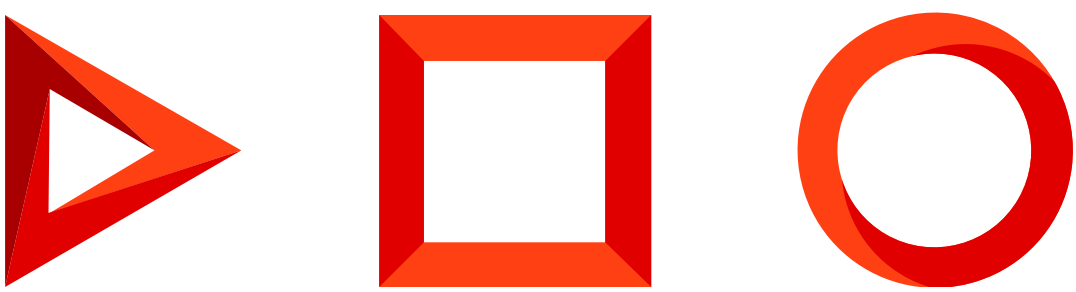

Table of Contents

Recommendations on app creation

PRODUCTS: ALL CREATIO PRODUCTS

Since version 8.0, Creatio follows a new customization approach and includes a new **app** level. Apps are function blocks that solve business problems. An app can consist of one or more content packages. Apps become the main unit of Creatio low-code development.

Creatio provides a wide range of app creation opportunities:

- Create apps from templates.
- Customize apps in a visual editor.
- Collaborate in development and testing environments.

Low-code tools streamline the app creation process and make it accessible for both professional developers and business users.

The app creation process involves several stages regardless of the app scale and purpose:

1. requirement collection and idea formulation
2. prototyping
3. development (using a visual editor or code)
4. testing
5. delivery
6. maintenance

The article covers each stage with emphasis on the best practices. You can use Service Creatio, enterprise edition to track app development, improvement, and support efforts. This product lets you manage changes, problems, and releases. Learn more in a separate guide: ITSM tools.

Requirement collection and idea formulation

Start working on the app by collecting requirements to determine the expectations of users from prospective functionality. To do this:

1. Formulate a concept and define a problem to solve or goals to reach.
2. Determine the channels for user feedback collection. For example, interviews, surveys, etc.
3. Record the collected user requirements in the most convenient format. For example, describe them as text, scenarios, user stories, business processes, etc.
4. Devise feature implementation and user delivery stages. Determine how the app MVP must look.
5. Note. MVP (Minimum Viable Product) is a test app version that provides value for users. You can use an MVP to collect feedback and expand the requirement list.
6. Organize the collected requirements as tasks and add them to a unified backlog, such as Jira.

Recommendations on requirement collection and recording:

- To create and prioritize tasks, use tools that support online access, co-editing, change versioning. For example, you can use Studio Creatio, free edition, Visio to describe the requirements as processes. Cloud test editors, Confluence, Jira, Miro are optimal for managing text descriptions. Use online tables or User story mapping format in Miro, Jira for prioritization.
- Update the list of requirements regularly, including the requirements based on user feedback.
- Prioritize the collected requirements based on the devised stages of functionality implementation.

Prototyping

1. Prototype the app in the early stages to collect user and customer feedback, test hypotheses, and evaluate the app UX before you begin the development. This strategy is time-efficient when the hypothesis is incorrect and generally good for the quality of solution development when the hypothesis is correct.
2. Before you prototype, collect the requirements, determine the end consumer, and explain the app context to the development team.
3. Start prototyping and create the visual design after that. To do this:
4. Generate articles, drafts, charts, object models of the new app that let you describe and visualize the app in as much detail as possible. Use Studio Creatio, free edition, Confluence, Miro, and other tools for this purpose.
5. Prototype the app logic using low-code tools. For example, you can create a descriptive process in Studio Creatio, free edition and convert it into an executable process to run a preliminary check.
6. Devise and implement the app UI using no-code tools in the Freedom UI Designer of Studio Creatio. This lets you create interactive prototypes quickly. You can use Axure, Adobe XD, Figma, or other visual editors for non-standard components and solutions.

Recommendations on prototyping:

- Select a tool depending on the problem and required level of detail (Adobe XD, Figma, Studio Creatio, etc.).
- Devise the app integration options and app position within Creatio architecture.
- Create a dynamic, not static, prototype and feature real app use cases in it.
- Test the prototype on the user group that corresponds to the real target audience. In this case, the test scenario must direct users without imposing correct solutions on them.
- Record, structure, and analyze the feedback for the app prototype. If you identify significant problems, stop the tests and revert to improving the prototype.

Development

App development involves the implementation of app logic and UI based on the approved prototype and visual design.

Usually, developers accomplish this task by coding.

You can create and customize apps in Creatio. No-Code Designer provides a unified UI that comprises various visual editors for UI, business logic, and integration setup, as well as process automation. Learn more in a separate article: Set up the app UI.

Recommendations on development:

- Use the MVP approach and divide app creation into small iterations/steps.
- Devise the development concerning deliverable improvements created as a result of each iteration/step.
- Use separate environments for development, testing, and production.
- Use a version control system for co-development.
- Analyze the existing solutions that solve your problem and use every option that the platform and low-code tools provide (base components and functionality, Creatio Marketplace solution) to speed up development.

Testing

The purpose of testing is to release a high-quality app that does not need rework. Check whether the app functions operate as intended as part of testing. To do this, log the detected errors and feedback.

It is equally important to check whether the app is usable and understandable by analyzing the scenarios that involve major features.

Recommendations on testing:

- Involve as many real users in testing as possible.
- Do not skip the testing stage even if the changes are minor.
- Run tests only in the testing environment.
- Test the app in conditions that most closely resemble reality. Run tests in various browsers, on various devices, etc.
- Test the app workflow for every needed role with the corresponding permissions applied.
- Review positive, negative, edge cases.
- Record the entire feedback, prioritize the error fixes and needed improvements.

Delivery

The functionality becomes available to end users at this stage.

After the deployment, evaluate the delivery by analyzing the fulfillment status of initial goals and the activity rate of users in the app. Later on, analyze the business-critical indicators and collect stats regularly.

Recommendations on app delivery:

- Deliver only thoroughly tested functionality.
- Make sure the delivery contains the needed artifacts (dependencies, data).
- Back up the database before you deploy the delivery.
- Accompany each deployment with a change and update log.
- Inform users of new deliveries.
- Schedule the delivery within the maintenance window.
- Perform deliveries as often as possible to reduce the volume of changes per delivery. It is a good idea to implement changes iteratively so that you can receive and process feedback for the changes immediately.

Maintenance

App management continues after delivery. It is important to maintain the released app throughout the entire life cycle.

Recommendations on app maintenance:

- Plan out training on changes for each delivery in addition to providing changelogs.
- Collect and process feedback regularly to fix errors, as well as improve the app and plan out further changes.
- Forward owner comments on requests and criticism to the users who submitted the feedback. Specify ETAs for error fixes in the comments.
- Prioritize the processing of the detected errors based on their severity and importance.
- Implement the improvements on and fix the detected errors as quickly as possible.