Creatio Marketplace

Development Guide

ACCELERATE

Table of Contents

How to start the development for Marketplace

Introduction

Creatio Marketplace is an online platform, where users can easily find and order a ready-made solution for their business. Marketplace is a point of contact between a customer and a developer. It enables potential customers to browse, select and purchase the needed partner solution.

Two types of partner solutions can be published on Marketplace:

- Application – a solution that extends the capabilities of the base Creatio products or vertical solutions having additional business value.
    - Connector – an application that extends the functional capabilities of the base Creatio products and serves the purpose of Creatio integration with external services and third-party applications.
    - Add-on – an application that supplements Creatio base products with new modules, configuration settings and system elements.
    - Software solution – an application developed based on Creatio products that fulfills a need in a specific industry and has its own business value.
- Templates – Creatio components preconfigured by third-party developers that can be used directly or as a template for creating new elements. These can be business processes, customs cases, dashboard items or interface settings. Also these can be examples of description and visualization of business processes and dashboards (not available in Creatio).

Learn more about the Marketplace and its elements from the "Partner solution release rules and regulations" part of the Marketplace developer registration page.

Upon registration on the Marketplace, you can access your personal Developer workspace. Use the Developer workspace to register and publish your applications in Creatio Marketplace.

Overview of the application development and publication

To develop and publish a Marketplace application, you will need to register and get access to the Developer workspace. After that:

1. Set up the developer profile. The developer profile must contain up-to-date information that will be displayed in the Marketplace showcase. Please refer to the "**Developer profile setup**" article for additional information.

2. Request development and test environments. The development of a custom solution should be carried out in the development environment, which is a separate Creatio site.

- Cloud development environment can be used if developers use built-in development tools. Please refer to the "**Ordering development site**" article for details.
- On-site development environment is mostly used by developers using third-party IDEs (Visual Studio, WebStorm, etc.) or developing in the file system.
  The process of deploying Creatio on-site is described in the "Deploying Creatio application on-site" article.

ATTENTION

Installation files of the Creatio products for deploying on-site can be downloaded via the links in the "Creatio products setup files" knowledge base article.

3. Create Marketplace application packages in the development environment. More information about packages, their structure and composition can be found in the "**Creating of application packages**" article.

4. Perform the application development. On this step, you develop the custom product functionality. Take into account the requirements and recommendations given in the "**Application publishing requirements**" and "**Designing product UX**" articles.

The Marketplace application can be a Creatio modification of any type. This can be a new section, integrated third-party service, etc. A detailed example of integration of a custom third-party service to a Creatio Marketplace application is covered in the "**Developing a simple application for Creatio Marketplace**" article. More complex examples can be found in the "**Application development examples**" article. Brief description of Creatio basic integration methods is available in the "**Integration with Creatio and public API**" article.

5. Test your application. Before you publish your product on the Creatio Marketplace, you need to make sure that it works on the test site. Please refer to the "**Testing a Marketplace Application**" article for further details on

transferring the developed package.

6. Select the license type and license settings. More information can be found in the “**Marketplace application licensing**” article.

7. Register the developed solution in the Developer workspace. Register the application in the Developer workspace for preliminary verification by the Marketplace support service. You can learn more about this process in the “**Application registration in the Developer console**” and “**Template registration**” articles.

8. Publish your application. Learn more about the process and the specifics of publishing a Marketplace application in “**Publishing applications**”.

Developer workspace

Developer workspace is a Marketplace section designed to manage applications, templates and Marketplace services.

Before you start registering your first application, populate the up-to-date information about you and your company. You can learn how to do it in the “**Developer profile setup**” article.

You need to deploy a Creatio application to be able to develop a Marketplace solution. Learn how to request a cloud-deployed Creatio application from a Developer Workspace in the "**Ordering development site**” article.

After you finish developing the application functionality or the Marketplace template, test it using a separate Creatio application. We recommend using the free 14-day Creatio trial to test the newly developed functions. Learn how to transfer a functionality to the test site in the “**Testing a Marketplace Application**” article.

As soon as you take a decision to publish an application or a Marketplace template, register it in the Developer workspace. You can learn more about this process in the “**Application registration in the Developer console**” and “**Template registration**” articles.

The final step is publishing the application or the template in the Marketplace. You can learn about the publication process from the “**Publishing applications**” article.

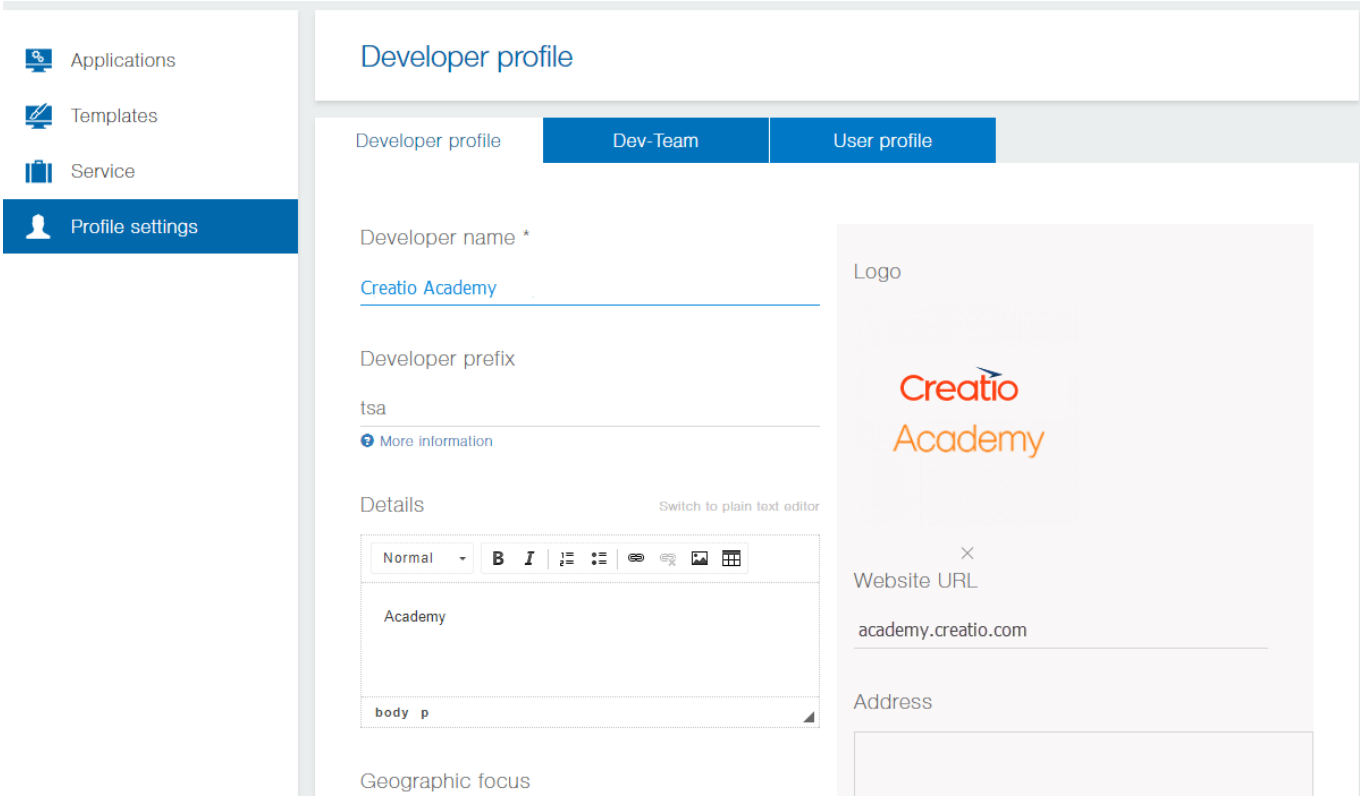
Contents

Developer profile setup

The developer profile settings should contain up-to-date information about your company. This information will be published as a separate page and displayed in your Marketplace showcase. This information must be entered before the first application is registered in the Developer Console.

To fill out information about the solution developer, go to [Profile settings] > [Developer profile] (Fig. 1).

Fig. 1. Developer profile



Fill in the following fields:

- [Developer name]* – the name of the developer company that will be displayed as the publisher of the application. The developer is a private entrepreneur, specify full name here. This field is required.
- [Logo]* – developer's logo, which will be displayed in the Marketplace showcase. We recommend using .png, .gif, .jpg, .jpeg images with white background, 200px in width.
- [Developer prefix]* – a unique ID added to the names of custom schemas, packages and custom columns in the objects inherited from the base objects, which enables you to determine the functionality created by the developer. The prefix identifies the functions added or modified by the developer. The prefix must be at least 3 characters long.
- [Details]* – brief information about the developer.
- [Website URL]* – developer's corporate website.
- [Developer contacts]* – specify developer's contact details for customer communication and providing support to those who download and use the developer's applications.
- [Support contacts]* – specify contact information of the developer's support service if any.
- [Geographic focus] – a continent, region or country to which the developer is oriented.
- [Industry specialization] – The main industry with which the developer works.

* – a required field.

Ordering development site

To be able to develop your own applications with Creatio platform, first you need to deploy the development environment - separate Creatio application which is used to create new functionality by developers. More details about working environments and Creatio development process organization are covered in the "Development process organization" article of the Creatio development guide.

You can deploy the development environment both in the cloud or on-site, using the local computer.

The advantages of using the on-site development environment are:

- involving special development tools (e.g., Visual Studio)
- using SVN
- using the development mode in the file system

The process of deploying Creatio on-site is described in the "Deploying Creatio application on-site" article of the Creatio user guide.

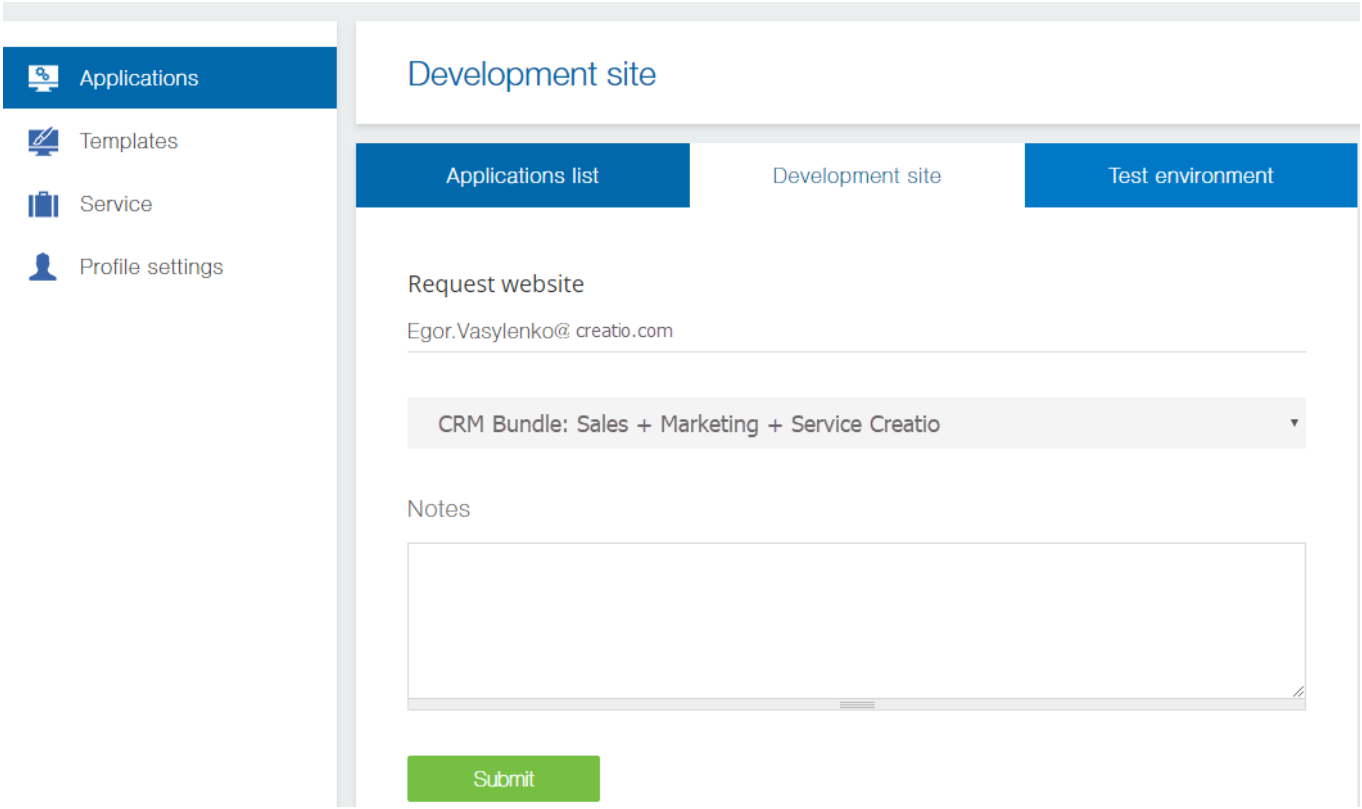
Cloud development environment is a bundle of the three flagship products: Sales Creatio, Marketing Creatio and Service Creatio. This allows you to use any of the basic functionality of Creatio in your own solutions development.

ATTENTION

Set the [Developer prefix] on the [Developer profile] tab in the [Profile settings] section before you order the development site. For more information please see the "**Developer profile setup**" article.

To deploy a new development site, use the site order form available in the Developer Workspace. To do this, go to the [Applications] section and submit a support service request on the [Development Site] tab (Fig. 1).

Fig. 1. — Development site order form



After the request is processed, the support service will provide you with your own cloud development environment within 8 working hours form the moment of sending your request.

When the development site is deployed, the user will be automatically notified to the email address specified in the order form. A link to the development site will be posted and available in the same tab from which the site was ordered (Fig. 2).

Fig. 2. — Link to the development site

Testing a Marketplace Application

Introduction

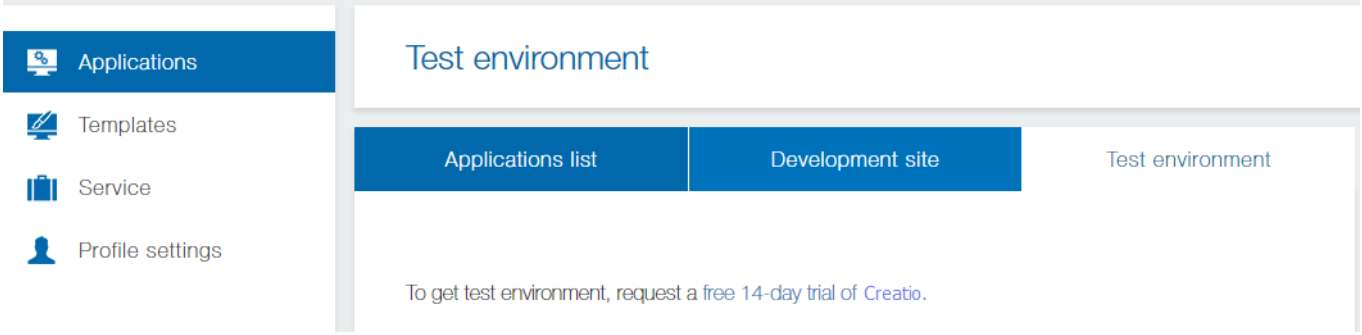
Development of complex functionality requires proper organization of the development processes. Recommendations on the development process organization are covered in the "Development process organization" article. Please refer to "Recommended development sequence" article for the sequence of development steps for deploying working environments in the cloud.

We recommend using the free 14-day trial of Creatio to test the newly developed functions.

Test site registration

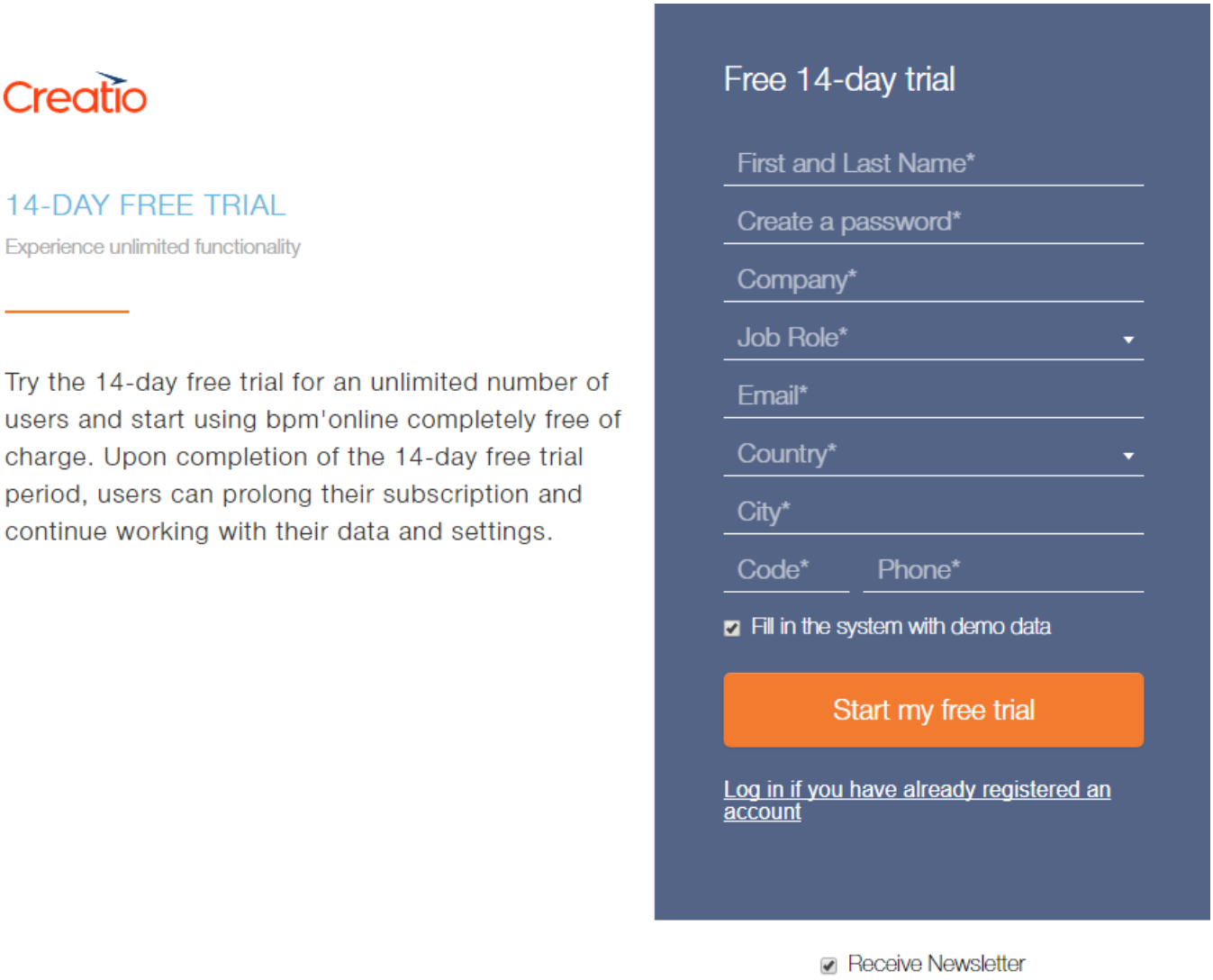
You can request a Creatio free 14-day trial for testing purposes directly from the Developer workspace. Go to the [Applications] section and select the [Test Environment] tab and use the link (Fig. 1) that will take you directly the trial request form.

Fig. 1. The free 14-day trial request link



Click the link, fill out the form (Fig. 2) and click [Get Started].

Fig. 2. Order form for the trial

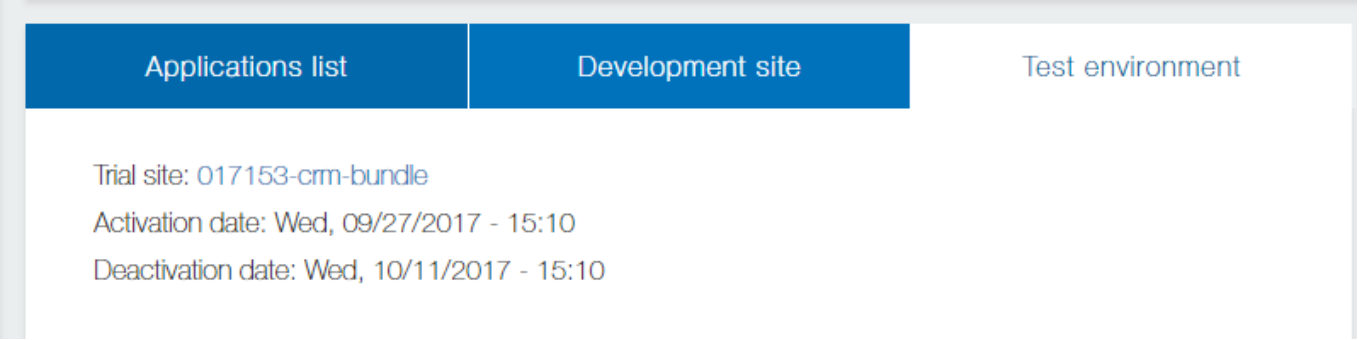


The trial is available immediately. The [Test Environment] will display the activation and the deactivation dates (Fig. 3).

NOTE

You will receive a notification of the readiness to use the test application on the email address specified at registration.

Fig. 3. Link to the test environment application



Transferring solutions to the testing environment

Before you publish your product in the Creatio Marketplace, you need to make sure that it works on the test site. To set up integration:

1. Export packages with the developed functionality from the development environment. For more information please see the "Exporting packages from the application interface" article.

2. Create a *.ZIP archive from the exported *.GZ package archives. If you exported a single package, there is no need to add it to a *.ZIP archive.

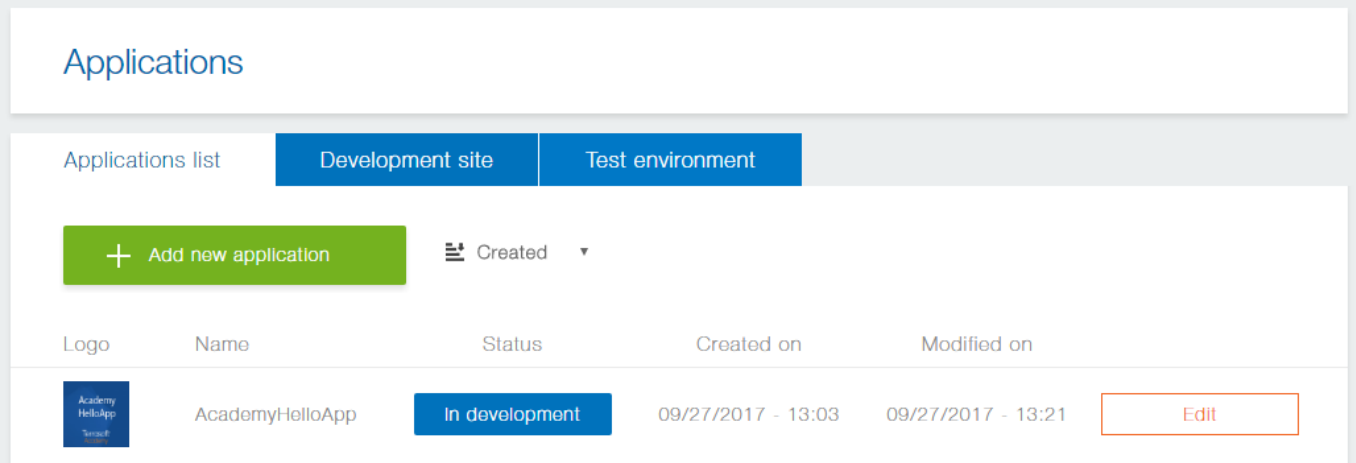
3. Import the *.ZIP archive (or the *.GZ file, if you exported a single package) to the test Creatio application. For more information on installing Marketplace applications in Creatio, see the following article: "Installing marketplace applications from a zip archive".

After the application has been successfully installed on the testing environment, please check your developed functions and make sure they work as intended. If any errors occur, please make necessary changes in your development environment, then repeat the export/install process again.

Application registration in the Developer console

New applications can be registered in the Developer Console on any development stage. Go to the [Applications] section and click [Add application]. After the registration is complete, your application will be added to the list in the [Applications] section.

Fig. 1. List of applications in the Developer console



The title with the application name and the application status are displayed on the application edit page. Status is the current stage of the application life cycle. An application can have one of the following statuses:

- In development – initial status of any application. It means that the product has not been released yet and is not available to the Marketplace users.
- Verification – the product has been submitted for publishing and is currently being reviewed by Creatio Marketplace support. One of the verification process stages is checking whether the loaded application packages and the base Creatio packages are compatible. All compatibilities specified for the product by its developer must be fully implemented and tested.
- Modified – the product modifications have been submitted for publishing by the developer and are currently being reviewed by Creatio Marketplace support. One of the review stages is making sure that the changes implemented by the developer are compatible with the primary requirements to applications.
- Published – the product has been verified and is now available in the Creatio Marketplace showcase.

NOTE

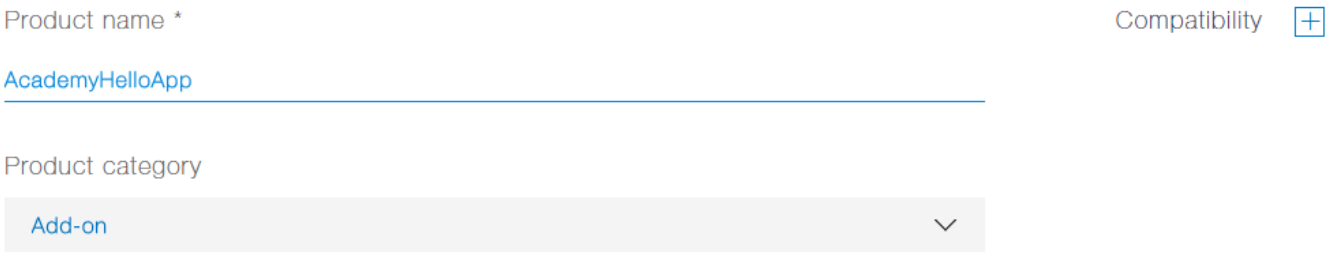
The application status is displayed after a new product has been registered. The "Clarification required" and "Published" statuses are set by the Creatio Marketplace support.

Application edit page contains several tabs with general properties of the application. Any changes made to the application properties page are available only for the developer and are not visible to the Creatio Marketplace users until published.

Introduction

The [General information] tab (Fig. 2) contains fields with general properties of your Creatio Marketplace application.

Fig. 2. The [General information] tab



The tab contains the following application properties:

[Product name]* – official name of the product on the Marketplace. For more information on how to properly name products for publishing them on the Marketplace, please see regulations on releasing partner solutions. This is a required property.

[Product category]* – category type of the provided solution, connector, add-on or a vertical solution.

Vertical solution – a custom configuration developed on the Creatio platform. Solutions fulfill the need of a specific industry and have a separate business value. A vertical solution consists of the base product used as a development platform and an Add-on developed by the partner to form the unique value of the vertical solution. More information about vertical solutions is available in the regulations on releasing partner solutions.

ATTENTION

Vertical solutions can be developed and published exclusively by organizations that have the Creatio Partner status.

Application– a solution that extends functions of Creatio base products and creates additional business value for customer. Applications for Creatio can be developed and published by any organization or individual, including organizations that do not have Creatio Partner status.

The applications fall into two subcategories:

- *Connector* – an application that connects Creatio with external services and third-party applications
- *Add-on* – an application that supplements Creatio base product with new modules, configuration settings and system elements

NOTE

To be able to use connectors and add-ons, users must have at least one Creatio license.

[Deployment option]* – deployment options that are available for the product (Fig. 2):

- Cloud – if product is deployed on Creatio servers
- On-site – if product is deployed on customer's servers

[Localization] – the list of languages that are available in the product. You can select multiple languages.

[Demo link] – a link to the deployed Creatio application with the installed product.

[Product summary]* – brief description of the provided product. The product primary function and the tasks it solves. The summary volume cannot exceed 115 characters with spaces.

[Logo]* – product logo displayed in the Marketplace showcase. Corporate background image in PNG, GIF, JPG or JPEG format, 262x216 px. Preferably dark. Lower half of the image must contain developer's logo in white color.

* – a required field.

Details

On the [Details] tab, you have text fields for describing the developed Creatio Marketplace application (fig. 3). You also have fields for adding and setting up images (fig. 4).

Fig. 3. Text fields of the [Details] tab

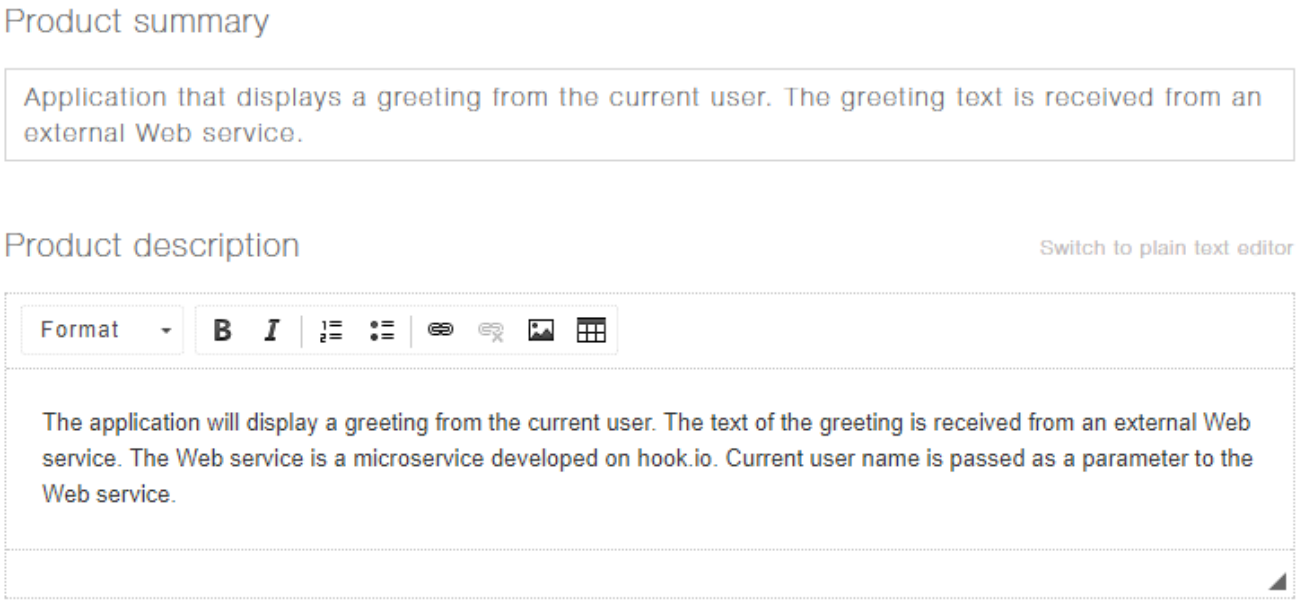
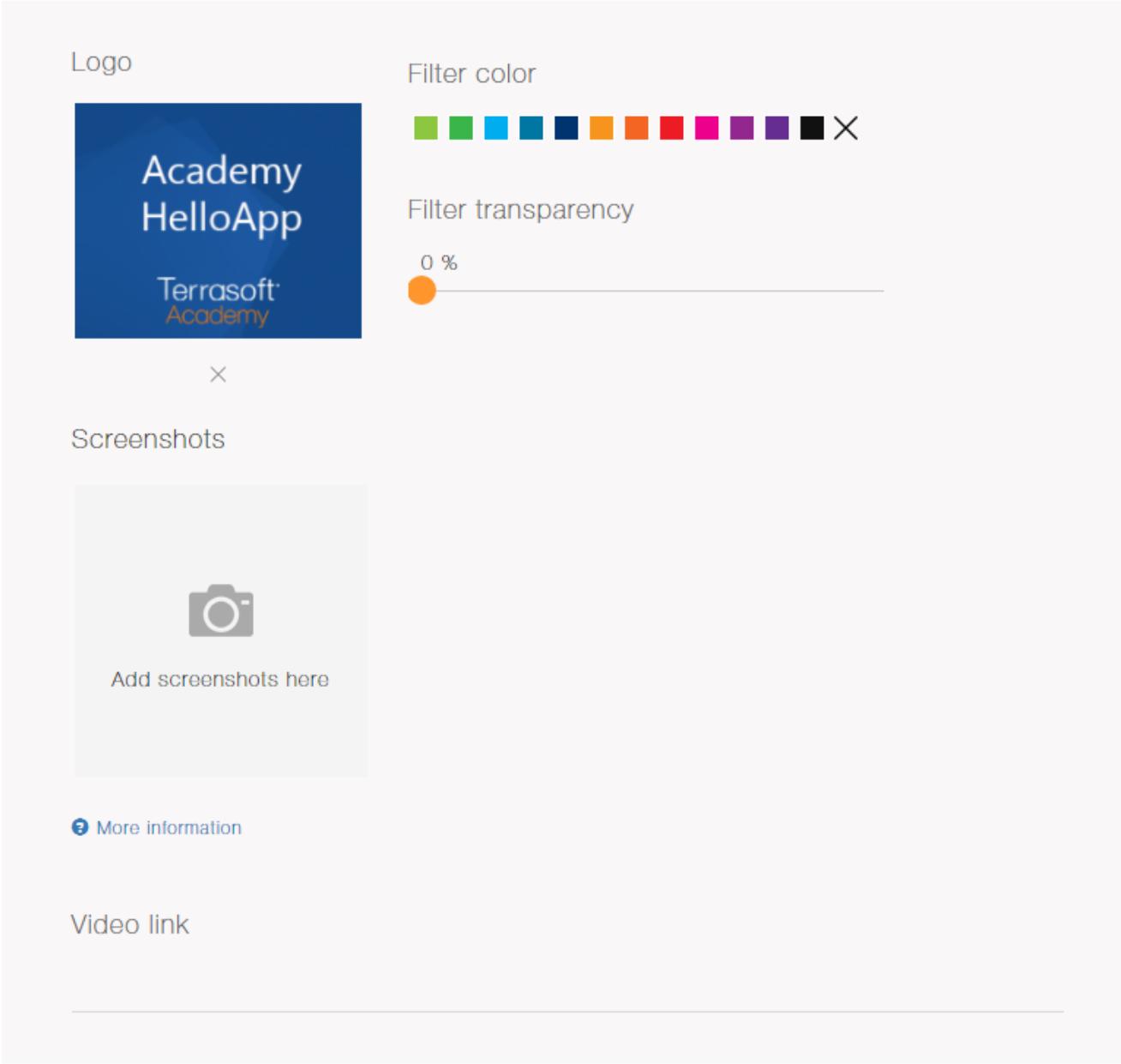


Fig. 4. Image fields of the [Details] tab



The tab contains the following application properties:

[Product description]* – extended description of the product. Complies with the primary Requirements and recommendations for writing descriptions to Marketplace applications and templates.

[Additional resources] – a hidden group of fields that enables adding links to third party resources or upload files with product description.

[Support conditions]* – terms and conditions of technical support that the developer obliges to provide for their product. The developer obliges to provide technical support within the boundaries of implemented functions by email, phone or any other communication channel.

[Screenshots]* – screenshots that illustrate the product and are available on the Marketplace product page. PNG, GIF, JPG or JPEG image, The minimum resolution is 1024px in width. and no larger than 20 MB.

[Video link] – a link to the product online video overview.

* – a required field.

Installation and setup

On the [Installation and setup] tab, you have information about the application installation and setup including the application compatibility with the base Creatio products. (Fig. 5, 6).

Fig. 5. Installation and setup of the application

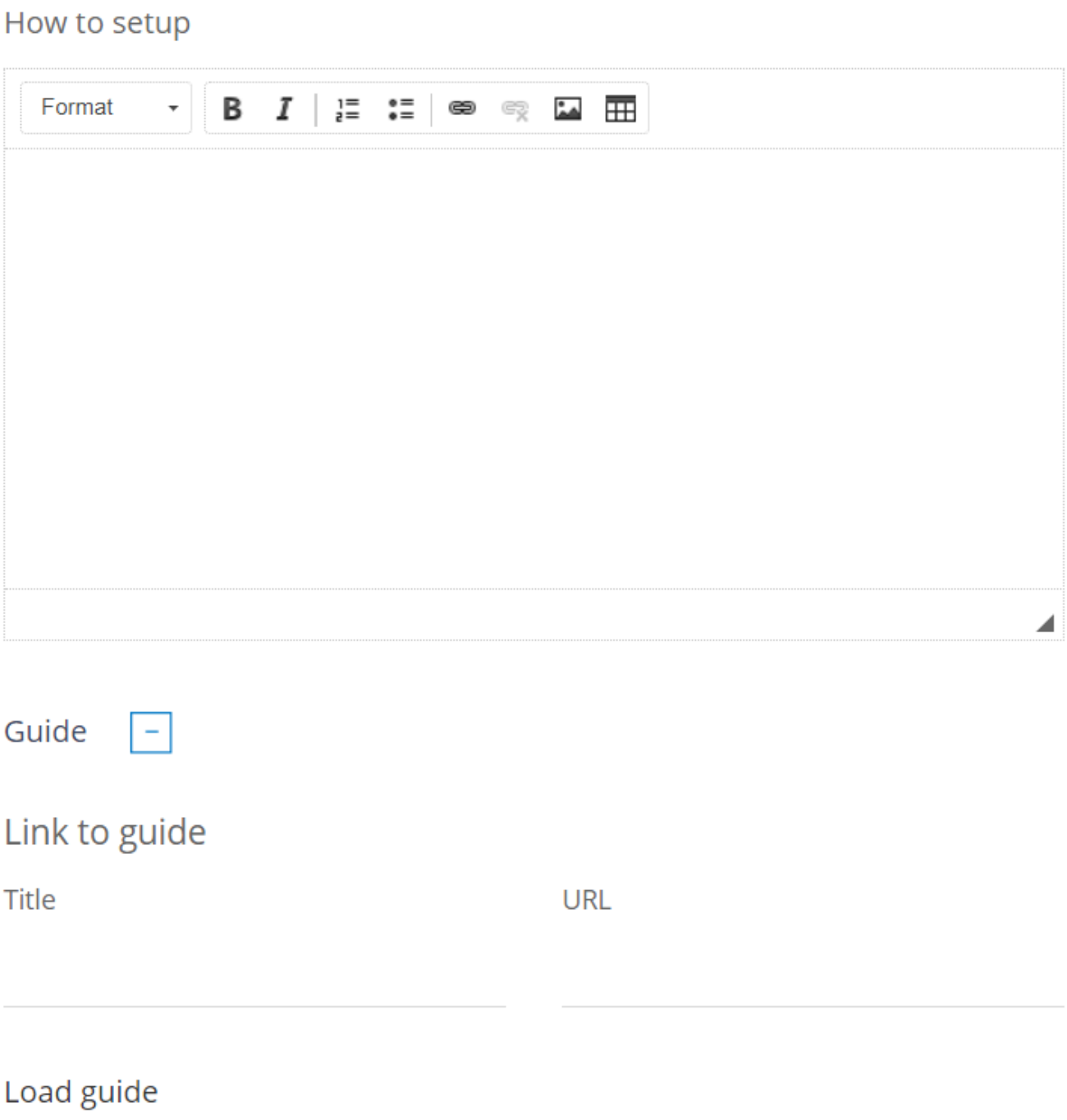
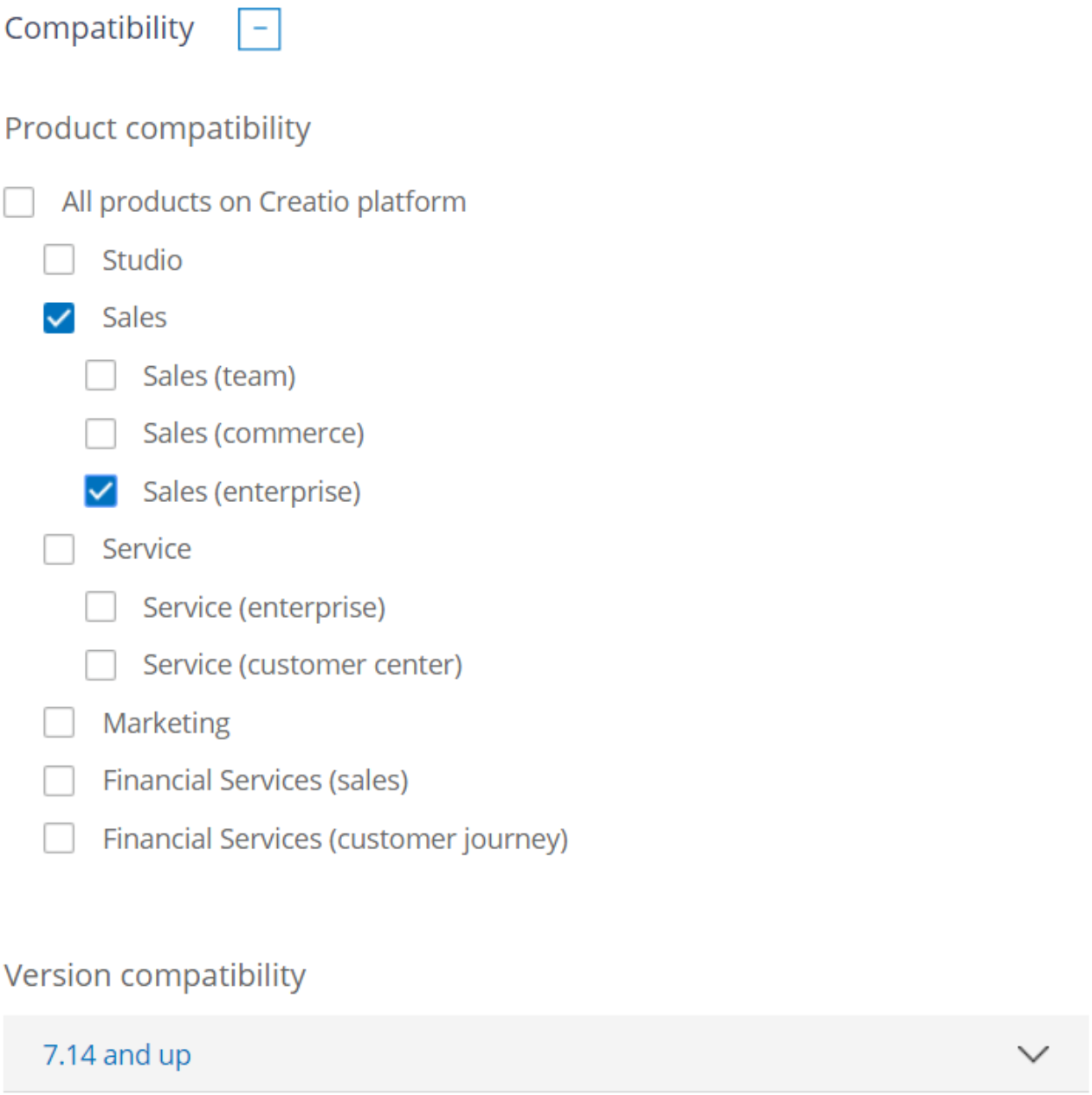


Fig. 6. Compatibility



The tab contains the following application properties:

[How to set up] – a step-by-step guide of the application setup by a user from scratch.

[Guide] – a hidden group of fields that enables adding links to third party resources or upload files with product description.

[Compatibility] – a group of fields with checkboxes for Creatio products. Select a checkbox to indicate that the corresponding product is supported by your application. In the [Version compatibility] field, specify the lowest supported Creatio version. Use the [Compatibility notes] field to specify any additional comments on the product compatibility, such as supported vertical solutions, etc.

* – a required field.

Packages and updates

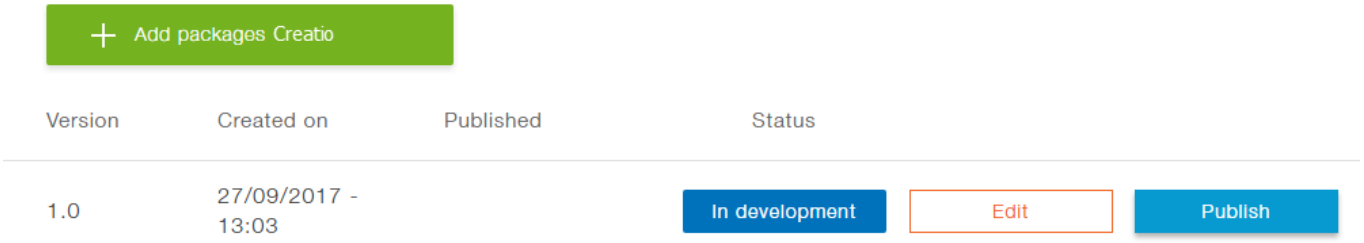
On the [Packages and updates] tab, you have information about all versions of packages and updates of the

developed solution (fig. 7). The version and the update status are displayed for each record. This tab also contains information about solution licensing, if it is not free of charge.

ATTENTION

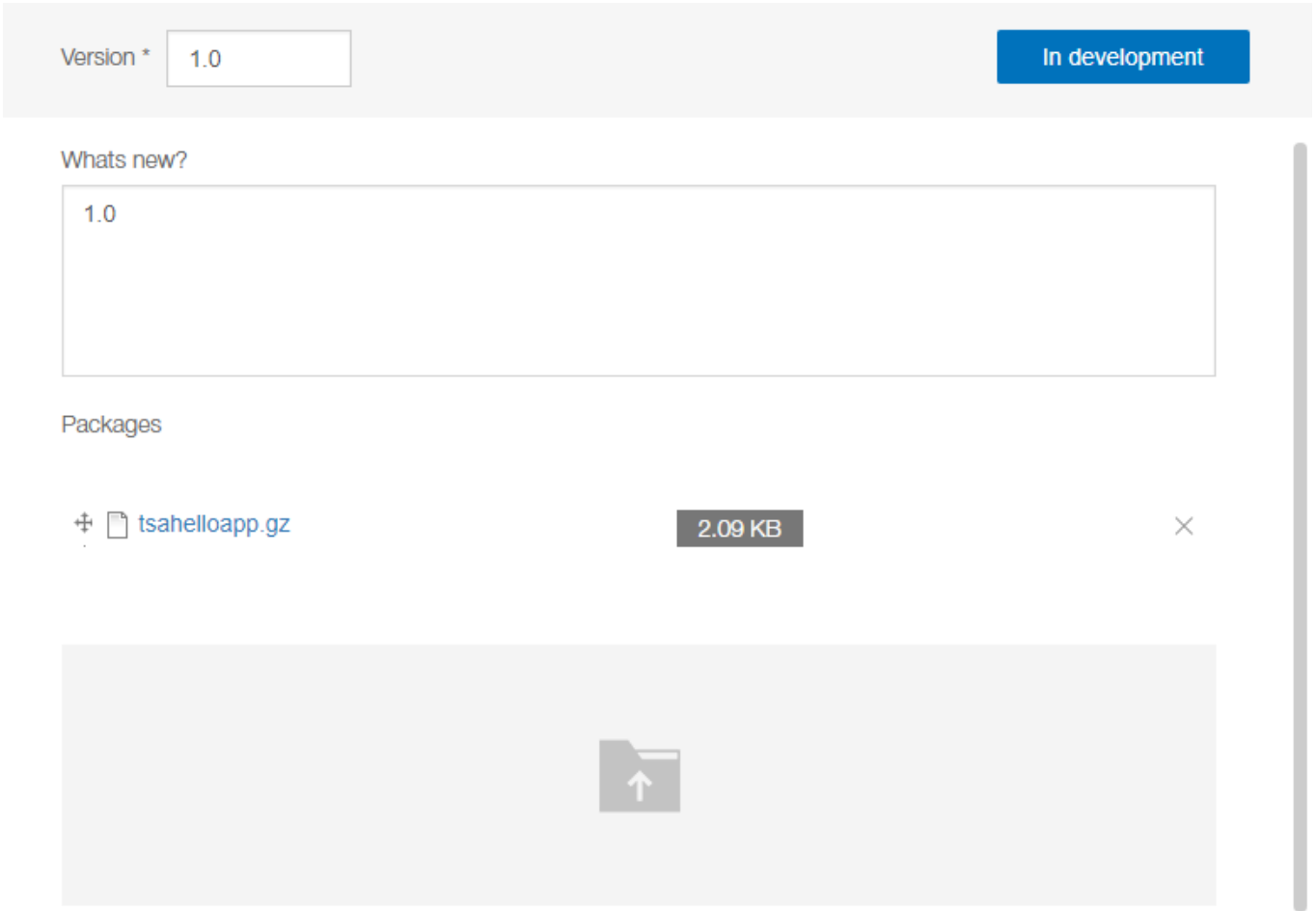
You cannot add a new application version before you publish the previous version.

Fig. 7. The [Packages and updates] tab



When you add a new package or edit an existing one, you will have the property edit page of the package or update displayed (fig. 8).

Fig. 8. Package property edit page



Package properties include:

[Version] – version of application, update or package.

[What's new] – brief description of the update.

[Packages] – list of uploaded packages. You can move the packages in the list or delete them.

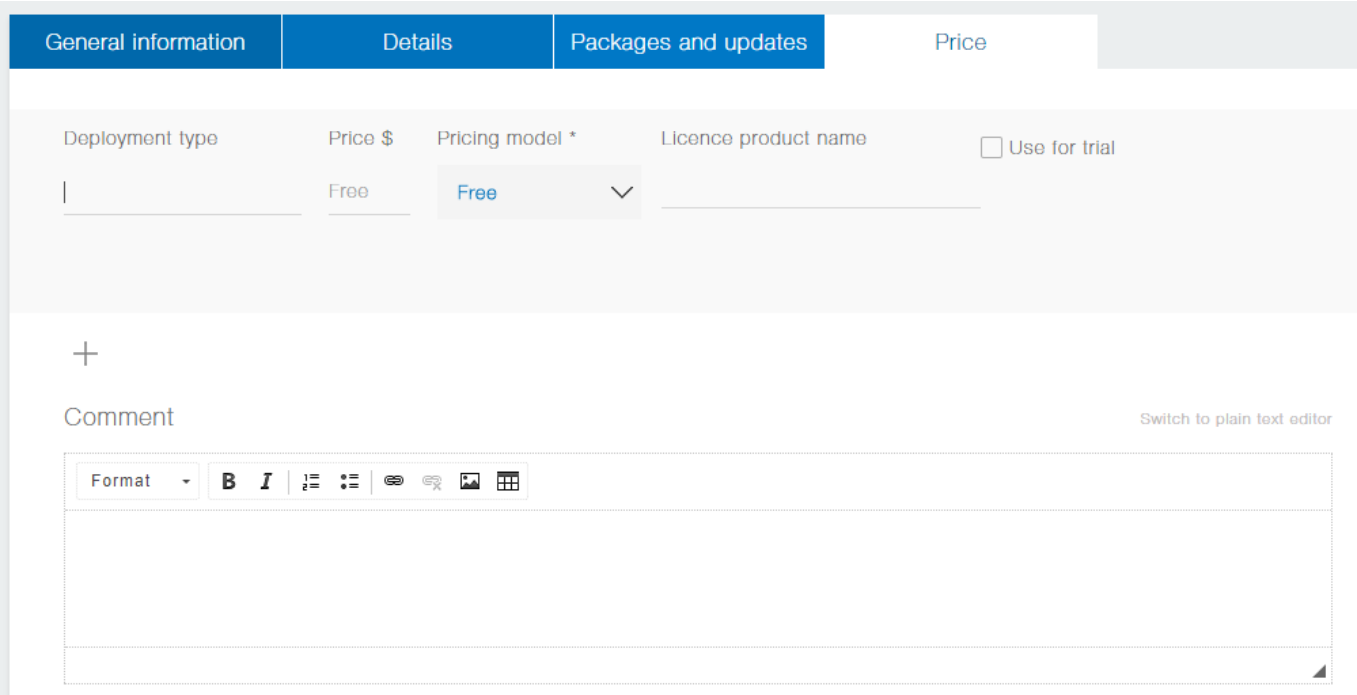
[Add Creatio packages] – package archive loading area. Possible extensions: *.gz, *.zip, *.rar. Maximum file volume: no larger than 20 MB.

[Manage licensing] – area for adding the licensed objects and operations, as well as selecting the license type.

Price

The [Price] tab (Fig. 9) contains fields for specifying different variants of price for the developed Creatio Marketplace application.

Fig. 9. The [Price] tab



The tab contains the following application properties:

[Base currency] – Marketplace base currency (US dollar is set by default). The currency displayed in the showcase for the end user depends on the domain of the web site that a user opens. Converting to national currency is performed according to Terrasoft commercial exchange rate.

[Product] – brief description of the provided product. For example: “Creatio sales enterprise cloud”.

[Price] – product price in US dollars. If the [Price] field is not populated, the product can be downloaded for free.

[Pricing model] – price format for the current product deployment type. The following price models are available:

- “/ year” – price for product per year regardless of the number of users
- “one-time” – customers can use the product for unlimited term after making a single payment, regardless of the number of users
- “/ user” – customers will need to pay once for every user who will be working with the product. The product can be used for unlimited term
- “/ user/year” – customers will need to pay annually for every user who will be working with the product. Customer pays for the subscription for one year and in a year must renew the subscription
- “on demand” – customers will need to make an inquiry with the developer to get their quotation.

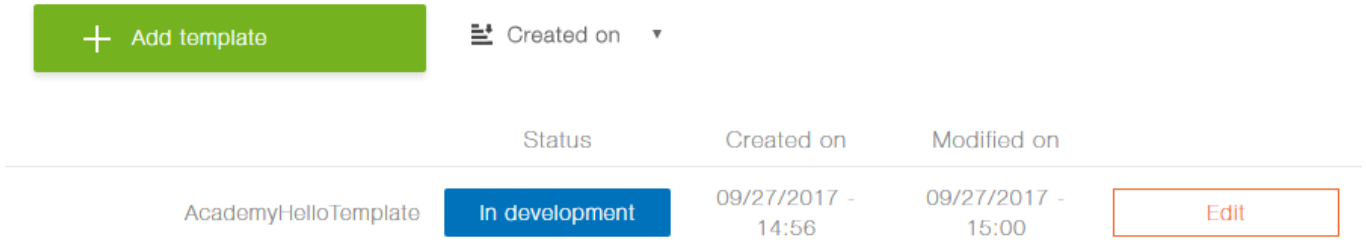
[Comment] – developer’s comment on the product pricing, that is visible to the Marketplace users.

Template registration

Templates are components of Creatio that are preconfigured by third-party developers, used directly or as an example for creating new elements. Business processes, custom cases, analytical elements and interface configurations can all be considered templates. Additionally, templates may be description and visualization examples of business processes and analytics, which are not executed in Creatio.

New templates can be registered in the Developer Console on any development stage. Go to the [Templates] section and click [Add template]. After the registration is complete, your template will be added to the list in the [Templates] section (Fig. 1).

Fig. 1. List of templates in the Developer console



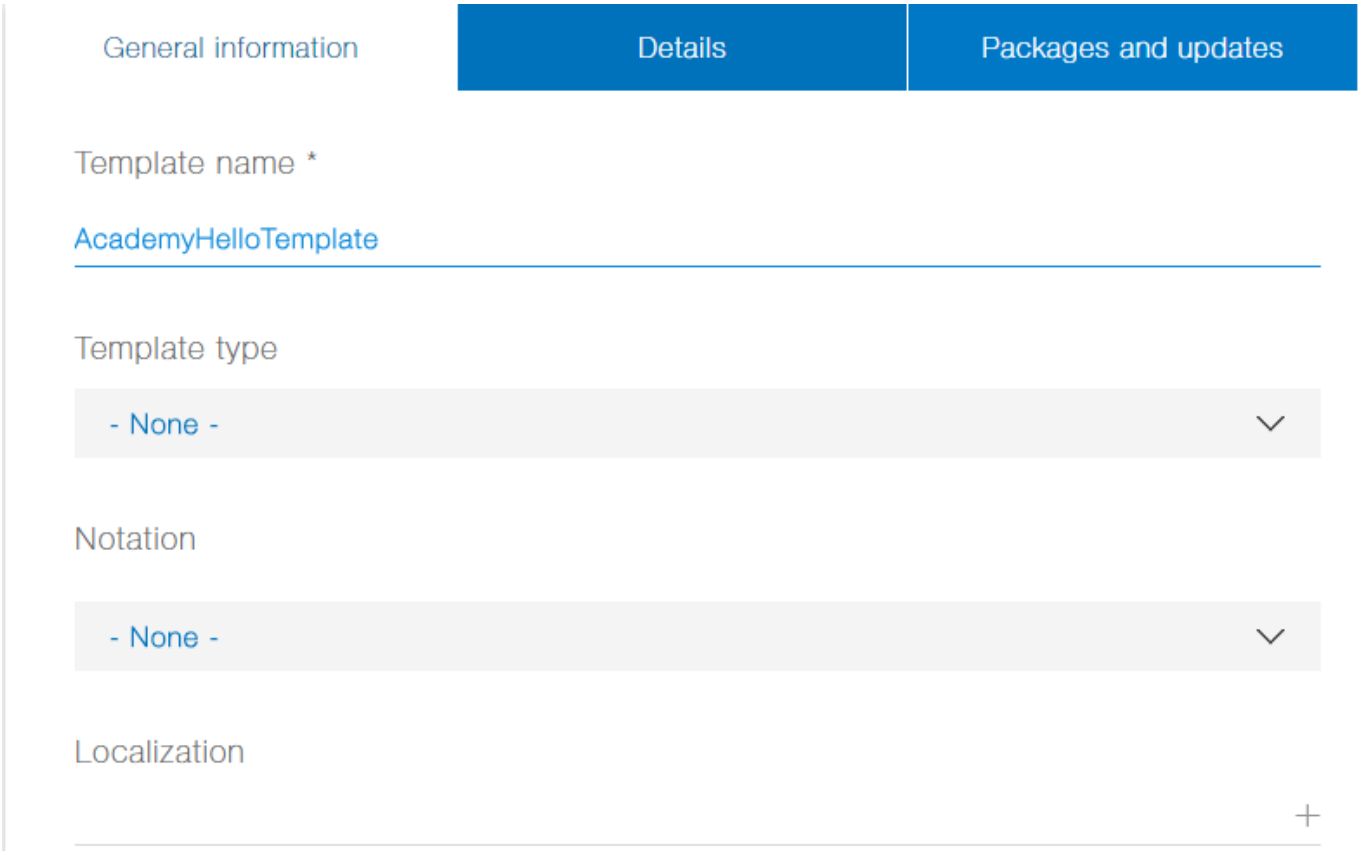
The title with the template name and the template status are displayed on the edit page. A template may have one of the following statuses:

- In development – initial status of any template. It means that the template has not been released yet and is not available to the Marketplace users.
- Verification – the template has been submitted for publishing and is currently being reviewed by Creatio Marketplace support. One of the verification process stages is checking whether the uploaded packages and the base Creatio packages are compatible. All compatibilities specified for the template by its developer must be fully implemented and tested.
- Modified – the template modifications have been submitted for publishing by the developer and are currently being reviewed by Creatio Marketplace support. One of the review stages is making sure that the changes implemented by the developer are compatible with the primary requirements.
- Published – the template has been verified and is now available in the Creatio Marketplace.

Introduction

The [General information] tab (Fig. 2) contains fields with general properties of your template.

Fig. 2. The [General information] tab



The tab contains the following template properties:

[Template name]* – official name of the template on the Marketplace. For more information on how to properly name products before publishing them on the Marketplace, please see regulations on releasing partner solutions.

[Template type] – the type of the template. Possible values are either “Analytics” (for analytical elements) or “Business process” (for business processes and cases).

[Notation] – business process notation, on which the templates is based. This field is only relevant to templates with the “Business process” type. Processes that are executed in Creatio only use DCM and BPMN notations.

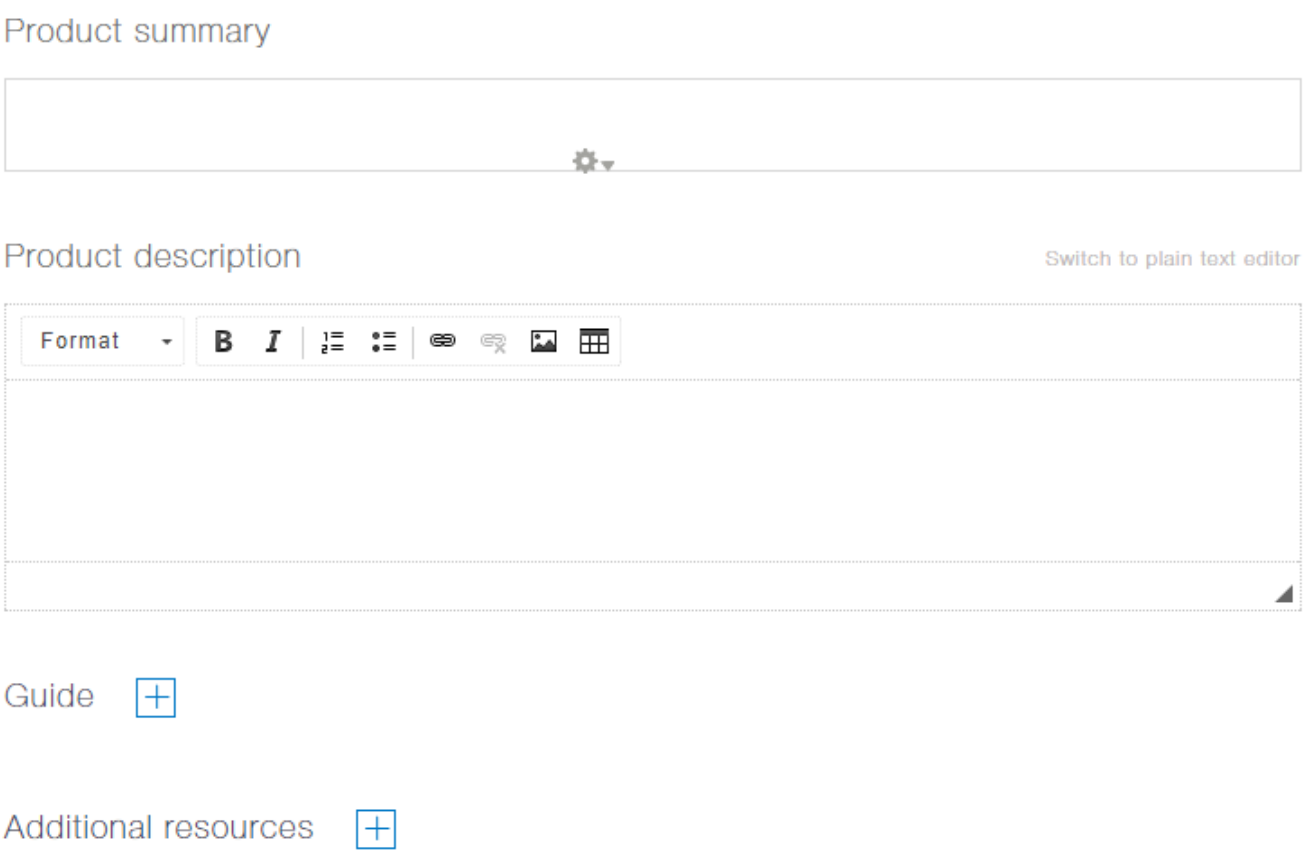
[Localization] – the list of languages that are available in the template. You can select multiple languages.

* – a required field.

Details

On the [Details] tab, you have text fields for describing the developed Creatio Marketplace template (Fig. 3).

Fig. 3. The [Details] tab



The tab contains the following properties:

[Product summary]* – brief description of the provided template. The primary function of the template and the tasks it solves. The summary volume cannot exceed 115 characters (with spaces).

[Product description]* – extended description of the template. Complies with the primary requirements and recommendations for writing descriptions to Marketplace applications and templates.

[Additional resources] – a hidden group of fields that enables adding links to third party resources or upload files with product description.

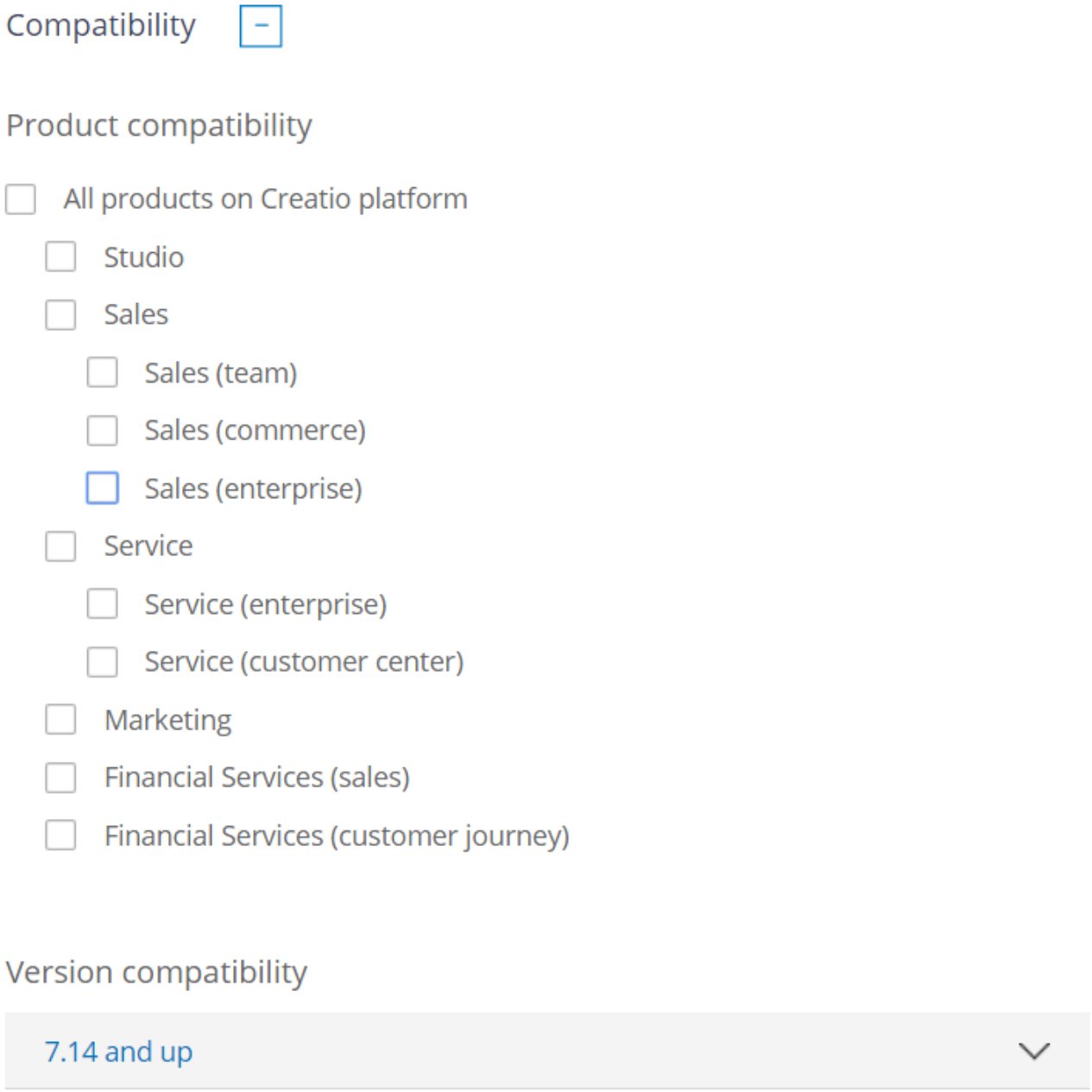
[Logo]* – template logo displayed in the Marketplace showcase. Corporate background image in PNG, GIF, JPG or JPEG format, 262x216 px.

[Screenshots]* – screenshots that illustrate the template and are available on the Marketplace product page. PNG, GIF, JPG or JPEG image. The minimum resolution is 1024px in width. The image should be no larger than 20 MB.

Installation and setup

On the [Installation and setup] tab, you have the information about the compatibility with the base Creatio products. (Fig. 4).

Fig. 4. [Installation and setup] tab



The tab contains the following template properties:

[How to set up] – a step-by-step user guide of the template setup.

[Guide] – a hidden group of fields that enables adding links to third party resources or upload files with product description.

[Compatibility] – a group of fields with checkboxes for Creatio products. Select a checkbox to indicate that the corresponding product is supported by your template. In the [Version compatibility] field, specify the lowest supported Creatio version. Use the [Compatibility notes] field to specify any additional comments on the product compatibility, such as supported vertical solutions, etc.

* – a required field.

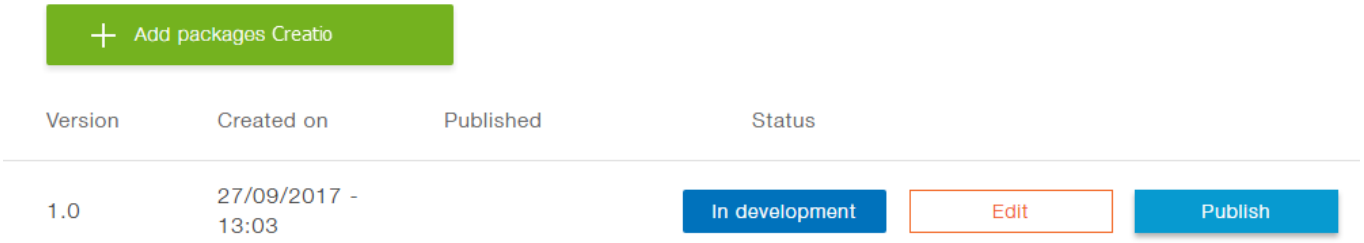
Packages and updates

On the [Packages and updates] tab, you have the information about all versions of packages and updates of the developed template (Fig. 5). The version and the update status are displayed for each record.

ATTENTION

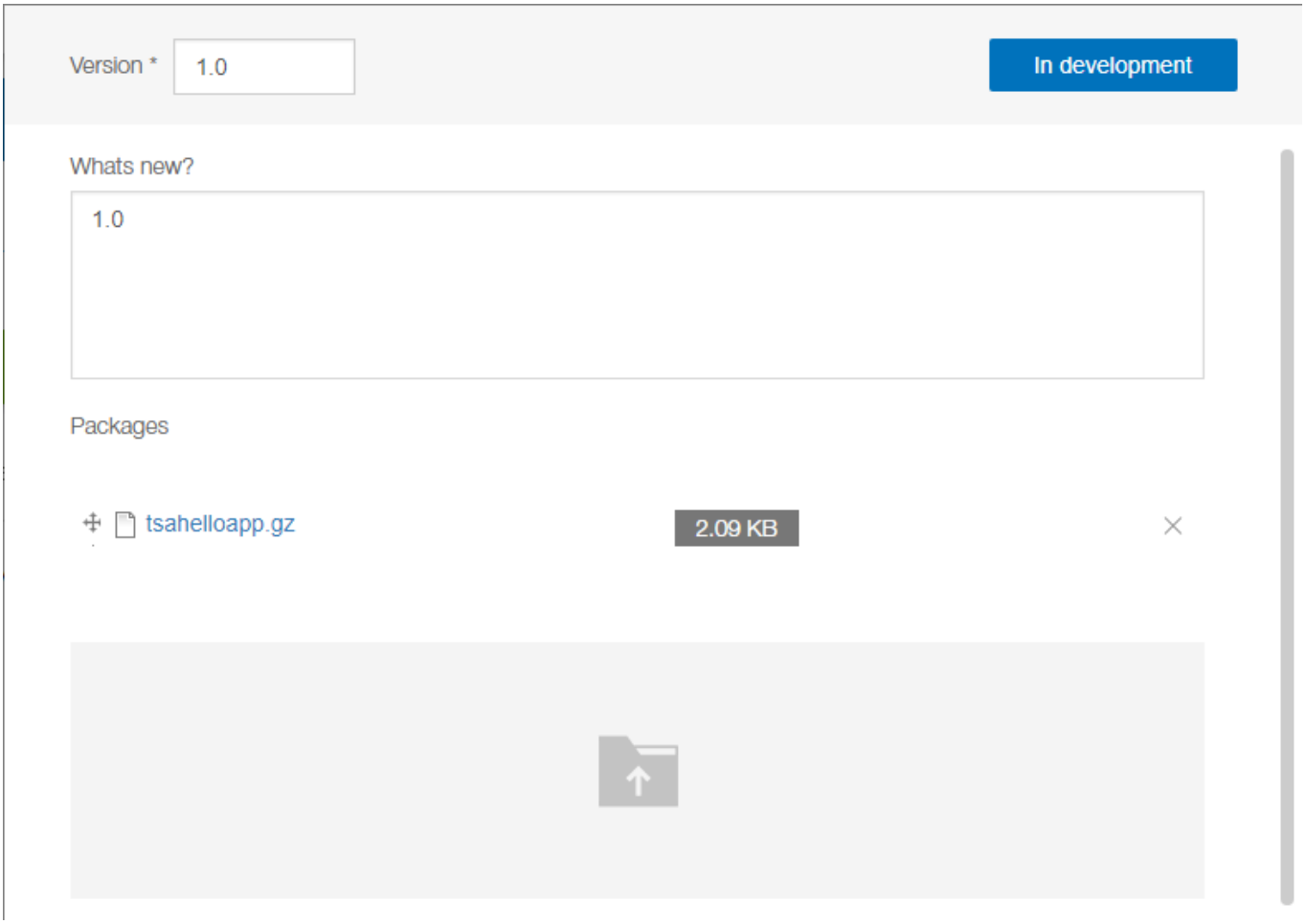
The new version of the template cannot be added before the previous version has been published.

Fig. 5. The [Packages and updates] tab



When you add a new package or edit an existing one, you will have the property edit page of the package or update displayed (Fig. 6).

Fig. 6. Package property edit page



The edit page contains the following fields:

[Version] – application, update or package version.

[What's new] – brief description of the update.

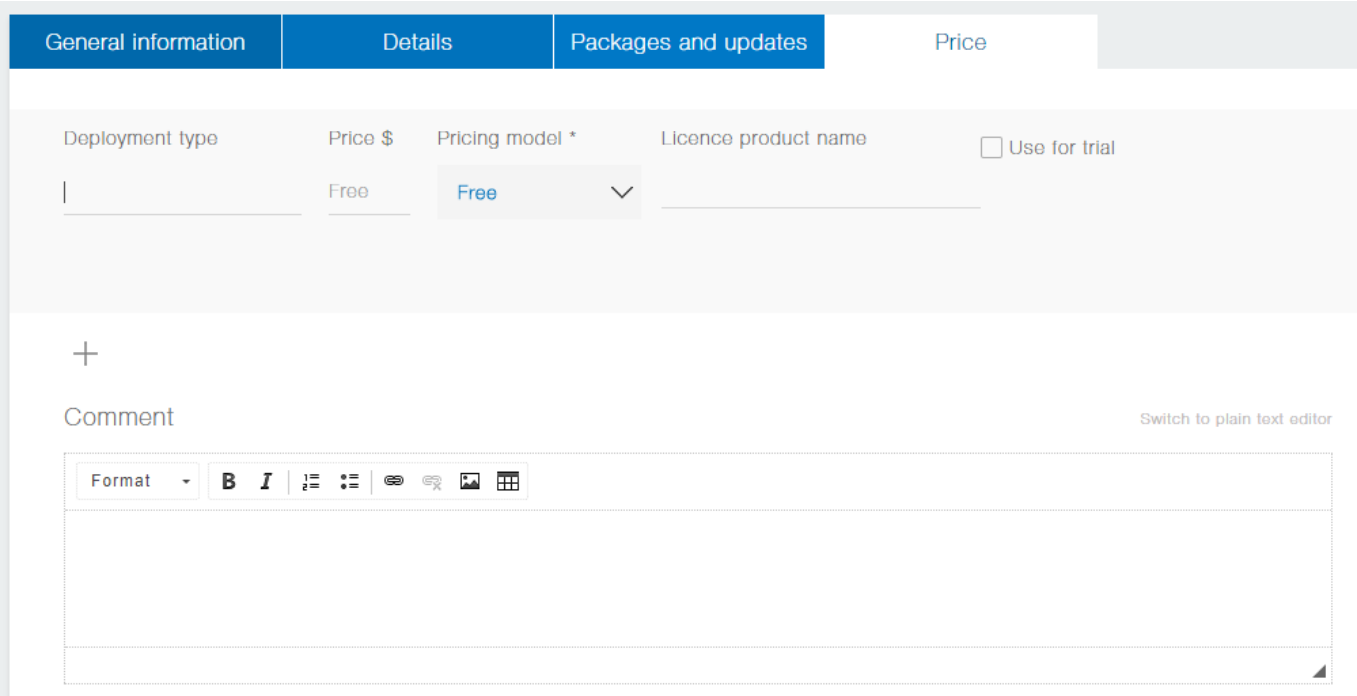
[Packages] – list of uploaded packages. You can move the packages in the list or delete them.

[Add Creatio packages] – package archive loading area. Possible extensions: *.gz, *.pdf. Maximum file volume: no larger than 200 MB.

Price

The [Price] tab (Fig. 7) contains fields for specifying different variants of price for the developed Creatio Marketplace application.

Fig. 7. The [Price] tab



The tab contains the following properties:

[Price] – product price in US dollars. If the [Price] field is not populated, the product can be downloaded for free.

[Format] – price format for the current product deployment type. The following price models are available:

- “one-time” – customers can use the product for unlimited term after making a single payment, regardless of the number of users.
- “on demand” – customers will need to make an inquiry with the developer to get their quotation.

[Comment] – developer’s comment on the product pricing, that is visible to the Marketplace users.

Publishing applications

Introduction

After you complete the development and testing of your application, you can start publishing it in the Marketplace. However, before you publish the product, study the regulations on releasing partner solutions. Make sure that the provided solution meets the **application publishing requirements**.

Publishing applications

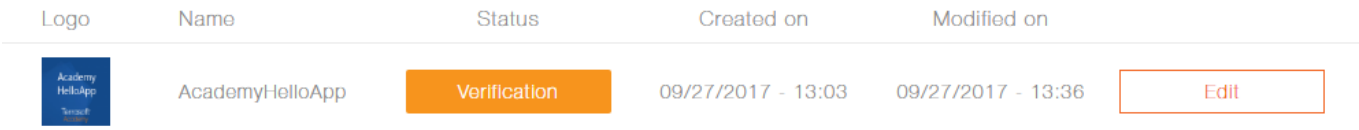
To publish the finished product, go to the application page in the Developer Workspace and click the [Publish]

button (Fig. 1).

Fig. 1. Initiating the publishing process

After the product is submitted for publication, it will be automatically sent to the Marketplace support service for verification. The product status will change to "Verification" (Fig. 2).

Fig. 2. Application status change



Product verification and its placement in the Marketplace

The verification process of the applications developed for Creatio Marketplace is required to ensure that the applications meet the **application publishing requirements**.

During the verification process, the Marketplace support service may send you comments to the published product. All comments will be sent to the developer using the Email specified in the developer's profile with detailed comments and recommendations for correction. The product status will change to “In development”. To complete the publication, correct all errors and re-submit the product by clicking the [Publish] button.

After the product is verified by the Marketplace support service, it will be automatically placed in the Marketplace showcase. The application status in the Developer Workspace will be changed to “Published”. After publishing, a corresponding notification will be sent to the developer by email.

Creating of application packages

Any Marketplace application is a specific set of packages, that are used to modify the configuration.

A Creatio package is a collection of configuration elements that implements particular block of functionality.

Learn more about configuration elements in a package and their structure in the ”Packages, schemas, modules” and “Package structure and contents” development guide articles.

Package dependencies, package hierarchy and main system packages are described in the "Package dependencies. Basic application packages” article.

When a customer develops new functionality and, therefore, creates new packages they have to use a revision control system (SVN). More information about creation of new packages, adding, installing and updating packages in SVN and exporting packages for transfer to other Creatio applications can be found in following articles:

- “Creating and installing a package for development”
- “Committing a package to repository”
- “Installing packages from repository”
- “Updating package from repository”
- “Exporting packages from the application interface”
- “Creating a package in the file system development mode”

Schemas are the main content of the packages. More information about adding schema to the package depending on its type can be found in following articles:

- Creating a custom client module schema
- Creating the entity schema
- Creating the [Source code] schema

Application publishing requirements

Basic requirements to the published applications

1. Product summary – a brief and user-friendly description of your application. The summary should cover the product's main functionality and the tasks that the product solves. Cannot exceed 115 characters with spaces.

2. Detailed product description – full description of your product capabilities and advantages.

3. Screenshots – at least 1 screenshot that illustrates the product's functions. The minimum resolution is 1024px in width.

4. Compatibility – Creatio products and versions, that your product is compatible with.

5. Price – specification of the pricing model for your product.

6. Packages – packages with implemented functions that are specified in the product description An application can be a single Creatio package (*.GZ file) or an archive (*.ZIP file) with several packages.

7. Application logo – a corporate background image in PNG, GIF, JPG or JPEG format, 262x216 px. It is not recommended to use white color for the company logo. Dark tones are preferred. Lower half of the image must contain developer's logo in white color.

8. Developer logo – a PNG, GIF, JPG or JPEG image, 200px in width, preferably with white background.

Marketplace application requirements:

1. Operability – the product must work exactly as claimed in the description.

2. Deployment – the application must be loaded in the Developer worspace on the Marketplace, in the [Installed applications] section. An application can be a Creatio package (*.GZ file) or an archive (*.ZIP file) with several packages. Before uploading your application, make sure that:

- A single file is uploaded.
- The uploaded file (package or ZIP) contains all functions required for proper operation of your application on supported Creatio configurations. I.e., additional setup of "base" or prerequisite packages is not required.
- Creatio package names remain unchanged and are the same as they were when packages were downloaded from the production environment.
- If a ZIP archive is downloaded, it cannot contain any folders. All *.GZ filed in the ZIP archive must be located in the archive's root catalog.

3. Compatibility – the product must be compatible with Creatio products and versions specified as compatible during registration. Complete the following sequence for all applications:

- Deploy a free 14-day trial Creatio application (available here: https://www.creatio.com/trial/creatio).
- Install the application from a zip archive. The setup procedure for Marketplace applications is covered in the "Installing Marketplace applications from a zip archive" article.
- After installing the application, log in to your Creatio trial. If errors were encountered during the setup, or you are unable to log in to Creatio after installing the application, it will not pass verification and cannot be published on Creatio marketplaece.

4. Licensing – unless distributed for free, Marketplace applications must have one of the available licensing models (see "**Marketplace application licensing**").

5. Performance – the product must not cause a performance decrease of the base Creatio product it was designed for.

6. Description – product description must not contain lexical, syntactic and semantic errors.

7. Data confidentiality – the Marketplace application must not obtain access to data to and transfer data illegally from the Creatio application where it is installed. Any data transfer that occurs after installation (in case of integration with third-party solutions) must be explicitly mentioned in the description and occur only after user authorization.

8. Support – the developer undertakes to provide technical support to the users within the implemented functionality by email, phone or any other communication channel. Support conditions must be explicitly stated in

the corresponding section of the product properties page.

9. Updates – when releasing Marketplace application updates, the developer shall include all bug fixes and modifications in the description.

Application certification requirements

Contents

Certification of Marketplace solutions

Certification of solutions is a special program for the developers of Marketplace solutions. Within the frame of the program, Creatio certifies specific solutions that meet our highest quality standards, get high net promoter scores and can be recommended to the whole community. The certification process and conditions are described in the certification process regulations document.

Advantages of the certified solutions

Certified solution badge

The Marketplace showcase displays a special “Application is certified by Creatio experts” badge.

Active expert sales

Information about the certified solutions is included in the Creatio expert commercial materials and partner networks. Certified application features are included in the examination program for Creatio sales managers. Certified Marketplace solutions are recommended for active sale, i.e., they are offered to customers proactively and not only upon customer request.

Solution quality review

All certified solutions undergo Creatio expert review and receive recommendations, e.g., for error corrections and application improvements.

Solution support tools

To assist partners with the certified solution support, Creatio enables case registration on the partner portal and case escalation that involves the support team and corresponding managers.

Solution certification process

Any Marketplace developer can apply for the certification.

The certification process consists of 5 stages:

1. Developer certification request
    a. Developer application analysis
2. Approval of the certification request
    a. Analysis of the solution certification priority
3. Application quality certification
    a. Providing materials for the certification
    b. Checking the Definition of Done (DoD) for the solution
4. Support certification
    a. Transferring the solution support to the portal

   b. Notification on implementing the support standards
   c. Support audit
5. Regular audit
   a. Evaluating the net promoter score (NPS)
   b. Evaluating the case resolution satisfaction
   c. Case escalation analysis
   d. Support audit

The certification process and conditions are described in the certification process regulations document.

Updates of the certified solutions

When a new version is released, the certified solution should correspond to all valid quality standards and DoD. To certify the updated application, a developer should provide the same set of materials as they did for the initial certification of the solution. When publishing the update, the Marketplace team controls the DoD of the certified solution the way they do it for the products that undergo their first certification.

Criteria of the application code review

Seamless update

- Applications should extend the base functionality and not replace it; therefore, you should not replace the modules. When you need to override non-abstract classes, call base methods while replacing schemas.
- Users do not work with SQL constructions but use ORM instead.

Performance

- The number of objects stored in RAM, as well as the number of flows, are limited. They do not depend on the number of users or DBMS volume (per page layout and buffering).
- Creatio performs processing operations in the background if users do not require the instant result of processing for continuing their work.

Integrations

- The application is protected against bulk incoming requests of the third party systems using light-weight gates and queues.
- When the external service is unavailable, users should still be able to work in the system. The integration process, though, might remain unavailable.
- In specific cases of the unavailable Creatio service, the operation of integration logic should be provided.

Data volume

- Archiving mechanisms should be implemented for data created automatically (e.g., logging).
- When working with large data volumes in tables, the database should limit users' arbitrary data operations and force working with the optimized data structure, using indices and denormalization.

Documentation requirements

Installation and setup guide

- Introduction: function of the application.
- Technical requirements.
- Application installation specifics (list any settings required in addition to the standard installation procedure).
- Description of the setup steps needed to get started the application.
- Description of the setup steps for the correct operation of the application (e.g., configuring additional features, description of processes and procedures that handle data without users' participation, etc.).

User guide

- Introduction: function of the application, specifics of usage.
- Description of user operations (all the developed features and functions).
- Examples of use cases.
- Notes and restrictions.

File requirements

- Format: PDF.
- The file must include a title page and a table of contents in addition to its target content.
- The title page must include the application name and company name/logo.
- The contents must support automatic updates.
- Documentation titles must be as follows: “%Application name% installation and setup guide” and “%Application name% user guide”.
- Use this link to download the file template.

Stylistics

- The text must be logically structured. Permitted format elements include headings, subheadings, marked/numbered lists and the bold font.
- Use imperative verbs for step-by-step instructions.
- The instructions must be sufficiently detailed and include all steps necessary to reproduce the described procedure. Any elements of the functionality, such as object names, UI elements and software components referenced in the documentation must match those in the actual application.
- The terminology must match that of the Creatio documentation. The terminology must be consistent throughout the documentation. Examples of terms:
    - Section;
    - List;
    - Page, record page;
    - Detail;
    - Tab;
    - Field;
    - Group of fields;
    - Action area;
    - System Designer;
    - Communication panel.
- Use the “em dash” (Alt+0151) “—” in the texts. Use the “en dash” (Alt+0150) between numbers: “10–15”.
- Do not use blank spaces in the “etc.”, “e.g.” contractions.
- Spell the names of the keyboard keys, e.g., arrows, functional keys, case shift keys with capital letters Example: “Press ALT+F3”. Do not spell with capital letters any descriptive keys, e.g., “Windows”.
- Enclose captions of interface elements in square brackets. Example: “Click [New]”, “Populate the [Name] field”.
- Use quotation marks: “”.

Text layout

- Use the “Verdana” font throughout the text, grey color (RGB 89, 89, 89).
- The font size of the body text (including tables): 10 px.
- The font size of the subheadings: 14 px.
- The font size of the headings: 16 px.
- The font size of the picture captions: 9 px.
- Line spacing of the body text: 1.15.
- Line spacing of the headings (including picture and table captions): 1.5.
- Page margins: 2.5.

- Do not use forced hyphenation in the text.
- Do not use indent spaces on normal text paragraphs.
- First level indent space: 0.63. Second level indent space: 1.9.
- Remove any blank paragraphs.
- Continuous numbering used in pictures and tables.
- Use the “Full justify” alignment for the body text.
- Use the “Left” alignment for headings and subheadings.
- Blue color (RGB 100, 184, 223) used for the marked and numbered lists of the first level.

Picture layout

- Permitted image formats in the documentation: PNG or JPEG.
- The key interface elements (fields, details, areas) can be highlighted. Use marquee select (frame width – 2 px) to highlight elements on a screenshot. Maximum number of highlighted elements in a screenshot – 2. Do not use other hightlighting means (arrows, text, etc.) in screenshots.
- All screenshots have captions and numbers.
- The picture caption and number placed at the top of the picture. Font. Verdana, font size – 8, color – light-grey (RGB 150, 150, 150). Left alignment. Line spacing of picture captions: 1.5.
- Text must reference all the pictures.
- Picture layout: “In Line with Text”.
- Alignment: “Full justify”.

“Training guide” requirements

Training guides on the certified Marketplace solutions will be available on the Academy site as a separate Marketplace course. For example, you can use the following link: https://academy.creatio.com/trainings

To create a training module on one of the Marketplace solutions, the partner should prepare and provide the following materials:

- Video tutorial
- Training article (optional)
- Test questions
- Practical assignment (optional)

An example of the training module is available under the following link.

Video tutorial requirements

Purpose: teaching employees and customers how to work with the solution

General requirements:

- Function of the solution
- Business cases for using the application
- Examples of tasks to solve
- Details on how the functionality works

Duration: 3-10 minutes. If the video duration exceeds 10 minutes, divide the video into several separate videos.

Content: a screencast of the setup procedures and use cases with voice-over. You can use presentation slides to illustrate the conceptual part in the video.

Use this link to download the slide templates (title+presentation).

Requirements to the resulting file:

- Video in the .mp4 format, extension 1920x1080.
- Voice-over script (may be used to add subtitles).

Requirements to the picture in frame:

- All texts appearing in the frame must be translated into the language of the current video. E.g., English videos should not contain texts in French or German and vice-versa. This applies to all texts in the frame, including browser captions or pop-up messages.
- Use Google Chrome (standard theme) browser to work with the application during screencasts.
- Do not display personal data on the screen (e.g., saved tabs, personal messages, etc). The author's name and picture, though, are permitted.
- You can use grey shading to highlight the clicks.
- Use standard Windows Explorer for working with the file manager.
- When working with the file manager, the folders must not contain any extra files that are not related to the demonstrated case.
- Cut out all technological moments, such as: application window switching UI, spelling errors when entering text, load times, loading masks, etc.
- To highlight the key interface elements (fields, details, areas), use the marquee select (a red frame with the width of 5 px) or the effect of blurring the rest of the area.
- Logical video blocks must be separated with title slides.

Training article requirements

Purpose: providing additional training materials on the aspects of the solution operation. The materials must be available in free access.

The article must contain:

- "See also" links for a deeper insight into the solution functionality.
- The links can reference Russian or English resources.
- The follow-up text should specify the resource, where the user will be redirected.

Duration of reading: 1-3 minutes.

Requirements to the resulting file:

- Format: *.docx.
- Volume: around 250 words (one industry-standard text page).

Requirements to the test questions:

Purpose: self-testing and revision of the covered material.

Volume: 3-5 questions.

If the solution involves several videos, there should still be a single list of questions for the whole solution.

Test duration: 1-10 minutes.

Requirements to the test questions:

- Include at least one marked correct answer and several incorrect answers with each test question.
- Multiple or single choice questions are permitted.
- The multiple choice questions contain an additional "Select all that apply" statement.
- The gist of the question must be clarified in the main video. The questions that are not explained in the video, difficult or "tricky" questions are not allowed.
- The question must be short and clear, no ambiguity permitted.

Requirements to the resulting file:

- A *.docx file that includes both the test questions and practical assignment.

Requirements to the practical assignment

Purpose: the final self-assessment for the participant, obtaining practical skills of working with the solution, if applicable.

Note that the website does not provide any means to verify or evaluate the assignment results and performance.

Content: the use case specifics (business case) must be unique, but the assignment logic should match the material provided in the video.

Duration: 15-20 minutes.

Requirements to the resulting file:

- A *.docx file that includes both the test questions and practical assignment.

Demo version requirements

Introduction

The purpose of demo versions is enabling a user to see how the solution operates.

A demo version is a set of demo data bound to a separate package that depends only on the solution package.

The recommended procedure of creating a demo version:

1. Deploy a clean Creatio build.
2. Install the solution package (e.g., *LabReports*).
3. Create a package titled as follows: *Package name + _Demo* (e.g., *LabReports_Demo*). Select the package of the installed solution in the dependencies.
4. Populate the sections and lookups with demo data, bind them to the demo package, sort the records to display the data correctly (follow the instructions below).
5. Export the package.

After you export the package, install it to a separate clean build and test the correctness and completion of its demo data. After you successfully complete the testing of the package with demo data, forward it to the Marketplace team.

For example, for the *Advanced excel reports for Creatio* solution:

1. Populate the records in the [Report constructor] section. When adding records, make different combinations of populating the primary fields for the object (e.g., [Report title], [Report view format], [Start with line], [Create column titles]).
2. Populate all details for several records (e.g., [Custom filters], [Report columns], [Section reports], [Report history]).
3. Add a template file for one of the populated records.
4. Populate records in the [Notifications] block, import the report.

Glossary:

- *Softkey data* – examples of populating the system objects. The softkey data is added to the solution primary package. The records should be in English and contain a "*(sample)*" text in their names.
- *Demo data* – examples of populating the system objects that are created in a separate package and depend on the primary package of the solution.
- *Showcase records* – section records populated with demo data. The showcase records are the first three records in a section.

Additional information on binding data to packages:

- Binding data to packages

Record author:

- All data are created on behalf of the same demo user bound by the user Id;
- Demo user (John Best) Id is 76929f8c-7e15-4c64-bdb0-adc62d383727.

Sorting and saving:

1. Set up sorting of lists to display showcase records on top. If the showcase records are not selected/populated, you can first select the sorting and afterwards populate the showcase records.
2. After you set the needed sorting, click [View] -> [Select fields to display]. Click the arrow on the [Save] button and select [Save for all users] option.
3. You can search for the needed record by the *Key* value that will match the sorting schema. You can view it in the *URL*. For example, sorting for the [Contracts] section will be saved in the record with *Key=ContractSectionV2GridSettingsGridDataView*. After saving for all users it will contain *ContactId=NULL*.

Specific/custom sorting must be saved in accordance with the above procedure. Sorting of the demo data cancels the softkey data sorting contained in schemas.

Populating sections:

- The number of records in sections should be sufficient to display on the screen and provide enough data for realistic analytics (about 20 records in average).
- Make sure that the number of table records does not exceed 1000.
- The first three records in a section (showcase records) must be the most complete.

Lookups

- Before you start populating sections, check and update the lookups.
- The lookups can contain softkey (bound to the primary package of the solution) or demo data.
- Do not duplicate records in lookups.

Record contents

- Record information must be consistent.
- Section records should be of different types. The record variety can be visible in folders and dashboards.
- The records must be logically bound to other records (we recommend populating the records in “chains” and not per section).
- Populate record information in sections, feed, notifications, email messages, comments, likes, etc.
- All demo data should have positive connotations. Even case incidents of the “complaint” category should not be clearly negative.

Record data

- Multiple system records contain a date field.
- To ensure that the record info is always up to date, use the script for shifting dates. It changes the dates based on the date of updating the demo records, specified in the “English Demo Data Actual Date” (ActualizedDemoDate_en) system setting.
- When you populate the demo data, change the dates in accordance with the system setting date.
- For example, if the update date is 01.11.2019, the script will change the 01.11.2019 date with the current date in the demo data. Likewise, the 01.10.2019 date will change to yesterday’s date, etc. A demo record whose date is 05.11.2019 will be recorded as a future date after the date shift, i.e., 4 months later than the current date of running the script.
- Dates can be added automatically when saving records. The build should not contain illogical dates.

The script for updating the dates:

```
/*
** Project: Creatio
** DBMS   : MSSQL 2008
** Type   : Script
** Name   : Actualize Date For Demo
*/
IF EXISTS (
    SELECT NULL
    FROM sys.objects
    WHERE [type] = 'TR' AND [name] = 'TR_BulkEmail_ProcessStatisticAfterUpdate'
)
DISABLE TRIGGER [dbo].[TR_BulkEmail_ProcessStatisticAfterUpdate] ON [dbo].[BulkEmail]
GO
IF EXISTS (
    SELECT NULL
    FROM sys.objects
    WHERE [type] = 'TR' AND [name] = 'TRCallAI'
)
DISABLE TRIGGER [dbo].[TRCallAI] ON [dbo].[Call]
GO


set nocount on
declare @NL char(1)
set @NL = char(13)

declare @ActualizedDemoDate datetime2
set @ActualizedDemoDate = (
    select top 1 sv.DateTimeValue
    from SysSettings ss, SysSettingsValue sv
    where ss.Id = sv.SysSettingsId
    and ss.Code like 'ActualizedDemoDate%'
    order by sv.ModifiedOn desc
)

if (@ActualizedDemoDate is null)
begin
    print 'DemoDate is null.'
    return
end

print 'DemoDate is ' + cast(@ActualizedDemoDate as varchar)
declare @DaysDiff int
set @DaysDiff = datediff(DAY, @ActualizedDemoDate, getdate())
declare @DaysDiffStr varchar(6)
set @DaysDiffStr = cast(@DaysDiff as varchar(6))
```

```
declare @Table sysname
declare @Column sysname

declare [c] cursor for
select table_name, column_name
from INFORMATION_SCHEMA.COLUMNS
where
    (upper(data_type) = 'DATETIME' or upper(data_type) = 'DATETIME2' or upper(data_type) = 'DATE')
    and (column_name not in ('ModifiedOn') or
            table_name in ('Opportunity', 'OpportunityInStage', 'SocialMessage', 'Lead', 'Contact',
'Account', 'KnowledgeBase', 'Product',
            'ApplicationApproval', 'Campaign', 'BulkEmail', 'CaseMessageHistory', 'Case', 'Activity',
'Invoice'))
    and (column_name not in ('CreatedOn') or
            table_name in ('Opportunity', 'OpportunityInStage', 'SocialMessage', 'Lead', 'Contact',
'Account', 'KnowledgeBase', 'Product',
            'ApplicationApproval', 'Campaign', 'BulkEmail', 'CaseMessageHistory', 'Case', 'Activity',
'Invoice'))
    and not table_name in ('SysLic', 'RemindInterval', 'SysRecentEntity', 'PlanYear', 'PlanMonth',
'PlanQuarter', 'Period', 'FinIndicator', 'CampaignPlanner', 'MktgActivity', 'CurrencyRate')
    and not (table_name in ('ContactAnniversary', 'AccountAnniversary') and column_name = 'Date')
    and not (table_name in ('Contact') and column_name = 'BirthDate')
    and not (table_name in ('SysAdminUnit') and column_name = 'PasswordExpireDate')
    and not (table_name in ('SysImage') and column_name = 'UploadedOn')
    and objectproperty(object_id(table_name), 'IsTable') = 1
    order by table_name, column_name

open [c]

while (1 = 1)
begin
    fetch next from [c] into @Table, @Column

    if @@fetch_status = -1 break
    if @@fetch_status = -2 continue

    declare @tempExecStr nvarchar(max)
    set @tempExecStr = (
        ' update [' + @Table + '] ' + @NL +
        ' set [' + @Column + '] = dateadd(DAY, ' + @DaysDiffStr + ', [' + @Column + '])' + @NL +
        ' where not [' + @Column + '] is null ' + @NL +
        ' and datediff(DAY, [' + @Column + '], ''99991231'') > ' + @DaysDiffStr)

    exec (@tempExecStr)

    set @tempExecStr = (
    'DECLARE @currentDate DATETIME2 = getutcdate();' +
    ' update [' + @Table + '] ' + @NL +
    ' set [' + @Column + '] = @currentDate' + @NL +
    ' where [' + @Column + '] > @currentDate' + @NL +
    ' and ''' + @Column + ''' in (''CreatedOn'', ''ModifiedOn'')')

    exec (@tempExecStr)

    print @Table + ' Column ->' +@Column + ' Diff->' + @DaysDiffStr + '(days) is updated.'
end
close [c]
deallocate [c]
GO
/* Actualize Date For Demo Forecasts */

IF (OBJECT_ID(N'ForecastSheet') IS NOT NULL)
BEGIN
    /*Actualize opportunity DueDate for demo Forecast*/
    --DECLARE @minOpportunityDueDate DATETIME2 = (SELECT MIN(o.DueDate) FROM Opportunity o);
    --DECLARE @newMinOpportunityDueDate DATETIME2 = DATEADD(MONTH, ((DATEPART(YEAR, GETUTCDATE()) -
2000) * 12) + DATEPART(MONTH, GETUTCDATE()) - 2, CONVERT(DATETIME, '2000.01.13', 102));
    --DECLARE @opportunityDueDateDiff INT = DATEDIFF(DAY,@minOpportunityDueDate,
@newMinOpportunityDueDate);
    --IF @opportunityDueDateDiff <> 0 BEGIN
    -- PRINT 'Will add ' + CAST(@opportunityDueDateDiff AS VARCHAR(10)) + ' days to Opportunity
DueDate';
    -- UPDATE Opportunity
```

```
    -- SET DueDate = DATEADD(DAY, @opportunityDueDateDiff, DueDate)
    --END


    DECLARE @entityForecastMap TABLE(
        ForecastEntityName NVARCHAR(50)
    );

    INSERT INTO @entityForecastMap
    VALUES
        ('AccountForecast'),
        ('ContactForecast'),
        ('LeadTypeForecast'),
        ('OppDepartmentForecast')

    DECLARE @sql NVARCHAR(max);
    DECLARE @tableName NVARCHAR(100);

    DECLARE c CURSOR FOR
        SELECT ForecastEntityName FROM @entityForecastMap
    OPEN c

    WHILE 1 = 1
    BEGIN
        FETCH NEXT FROM c INTO @tableName
        IF @@FETCH_STATUS = -1 BREAK
        IF @@FETCH_STATUS = -2 CONTINUE

        IF (OBJECT_ID(@tableName) IS NOT NULL)
        BEGIN
            SET @sql = '
                UPDATE f
                SET PeriodId = (
                    SELECT
                        inp.Id
                    FROM Period inp
                    WHERE inp.PeriodTypeId = p.PeriodTypeId
                    AND inp.StartDate = DATEADD(year, DATEDIFF(year, p.StartDate, GETDATE()),
p.StartDate)
                    AND inp.DueDate = DATEADD(year, DATEDIFF(year, p.DueDate, GETDATE()), p.DueDate)
                    )
                FROM '+ @tableName +' f
                JOIN Period p ON p.Id = f.PeriodId
            ';
            EXEC sp_executesql @sql;
        END;
    END;
    CLOSE c
    DEALLOCATE c
END;

GO

IF EXISTS (
    SELECT NULL
    FROM sys.objects
    WHERE [type] = 'TR' AND [name] = 'TR_BulkEmail_ProcessStatisticAfterUpdate'
)
ENABLE TRIGGER [dbo].[TR_BulkEmail_ProcessStatisticAfterUpdate] ON [dbo].[BulkEmail]
GO
IF EXISTS (
    SELECT NULL
    FROM sys.objects
    WHERE [type] = 'TR' AND [name] = 'TRCallAI'
)
ENABLE TRIGGER [dbo].[TRCallAI] ON [dbo].[Call]
GO
```

Comprehensive record views

- If activities are created during adding data, check the records in the activity schedule.
- If you need to display analytics in the Dashboards when populating the demo data, check the charts in each sections taking into account the update date.
- Some sections have pre-configured filters by Owner and time period ([Activities], [Invoices], etc.). When you populate the data,

make sure that the grid list contains records meeting the filter conditions.

Designing product UX

Introduction

The negative user experience (in installing, configuring and working with the product) is the one of the main factors that significantly reduce the probability of purchasing a product after a trial. Potential customers often perceive quality and useful products negatively of they require complex and time-consuming setup.

We have come up with a number of recommendations to help avoid typical errors with UX of product setup and user's first impressions.

Clear navigation in the system

Convenient and logically organized navigation will help a new user to find the application, navigate to the necessary function and use it. When designing the navigation, consider following:

1. Displaying a section in the workplace

If the application functions include a new Creatio section, it should be displayed in at least one workplace. A workplace is a set of system sections available to a group of users. When adding a new section, take into account that Creatio workplaces are typically connected to specific functional and organizational user roles (i.e., sales, contact center, administration, etc.).

Case: "Adding the [Chats] section"

Incorrect	Create separate workplace and add the [Chats] section to it
Incorrect	Do not include section in any of the workplaces
Correct	Include the [Chats] section in the [Sales] and/or [Marketing] workplace

2. Accessing settings from the system designer

All custom sections and blocks used for setup and configuration should available in the system designer. The users can access the system designer by clicking the  button at the top right corner of the application window from anywhere in the system. The links to setup and configuration functions in the system designer are grouped into several blocks. The currently available blocks and their functions are listed in Table 1.

Table 1. Blocks of system designer settings

Settings block	Usage
Import and integration	Contains links to setup functions of different types of integrations, as well as access to Excel import. Add links to connector setup pages here.
System setup	Contains links to setup functions that affect general system behavior. Add links to custom application settings and vertical solution settings here.

Modify the system designer by adding links to your custom setup functions, such as:

- Custom settings pages
- Custom setup sections
- Lookups used as setup medium.

3. Accessing settings from the section actions menu

The user experience is commonly formed as follows: user opens a section with new functions, starts studying it and only then understands that further work requires additional setup.

To help users navigate and perform necessary settings, add required action to the [Actions] button menu. Please note, that the section page (list of the records) and the page of specific record are the different pages. For each page,

the set of actions can be different and the [Actions] button is configured separately for the list of records and the record page. The action added to the [Actions] menu of the section will not be added in the [Actions] menu of the record page.

Case: “Implement transition to the custom login authorization page in the X external system (login/password) that cooperates with the [Integration] section implemented in the application”

✗ Incorrect	Implement separate settings page accessible only by a direct URL address.
✓ Correct	Add the “Configure the connection to the X external system” item in the [Actions] button menu of the [Integration] section.

Settings are simple and easy to find

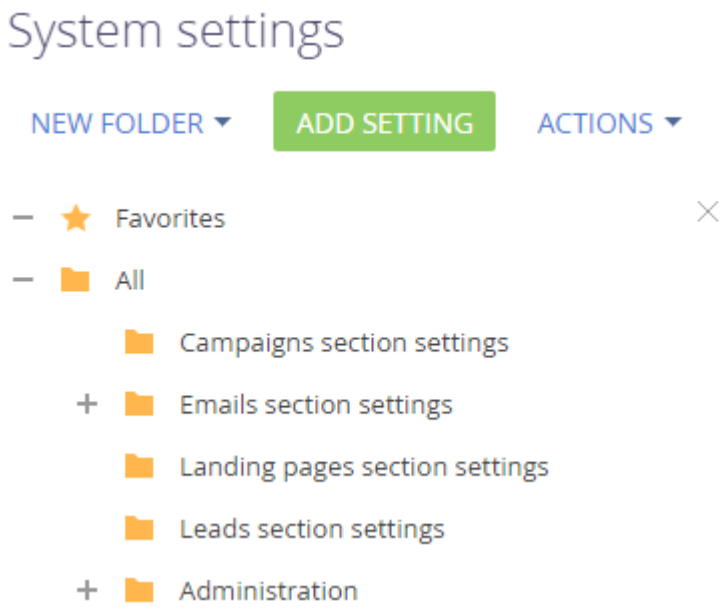
Many Marketplace applications require initial configuration by the user. To do this, the Creatio has a set of custom tools.

1. The [System settings] section

The [System settings] section is designed for additional setup of system sections. For example, set the color of overdue activities, parameters for sending emails, external service connection parameters, etc.

Case: “Add a new system setting”

Use short and clear names for system settings. The name should reflect the essence of the setting and point on functions to which it belongs.	✗ Incorrect (the name is too long): “The currency that is used in the system by default” ✓ Correct: “Base currency”. ✗ Incorrect (the name does not reflect the essence) “Order status”. ✓ Correct “Order status by default”. ✗ Incorrect (the name does not provide sufficient information about configured functions) “Automatic launch of the process” ✓ Correct “Automatic start of the incident management process”
Group the settings.	User has no chance to find a specific setting in the general settings list without knowing its exact name. Grouping of the settings is a simple and effective way to simplify application configuration.
Add the description	The description of the system behavior that the system setting affects will answer



most of user's questions.

2. The [Lookups] section

The [Lookups] section is used to manage all data (settings and system data) that can be displayed as a simple list.

The examples of using the [Lookups] section for different system tasks are given in the Table 2. It is highly recommended to use the lookups cases listed in the table.

Table 2. Lookup use cases

Task	An example of a lookup used
A list of data for user selection	[Document statuses] The lookup contains a list of all statuses that a document can have, for example, "Active", "Inactive" or "Draft".
Field autofill during integration	[Lead channels] Lookup contains the list of all resource types that provided new leads, for example, "Social accounts", "Search-based advertising" or "Email".
Configured business logic	[Case notification rule] Lookup contains a list of rules that manage sending notifications to contacts about their case progress (for example, "Send immediate ", "Send after a delay", "Disabled").
System list	[Webitel users]
Configuring the displaying of functionality in different sections	[Gantt chart configuration]

3. Settings in the user profile

Individual settings of a specific user (individual login/password or preferences for using certain product functions) can be added in the user profile.

Examples:

1. Integration connection settings:

- Id of the phone operator/phone number by which the user works with a phone integration connector.
- Settings for connecting the user to the mail service.

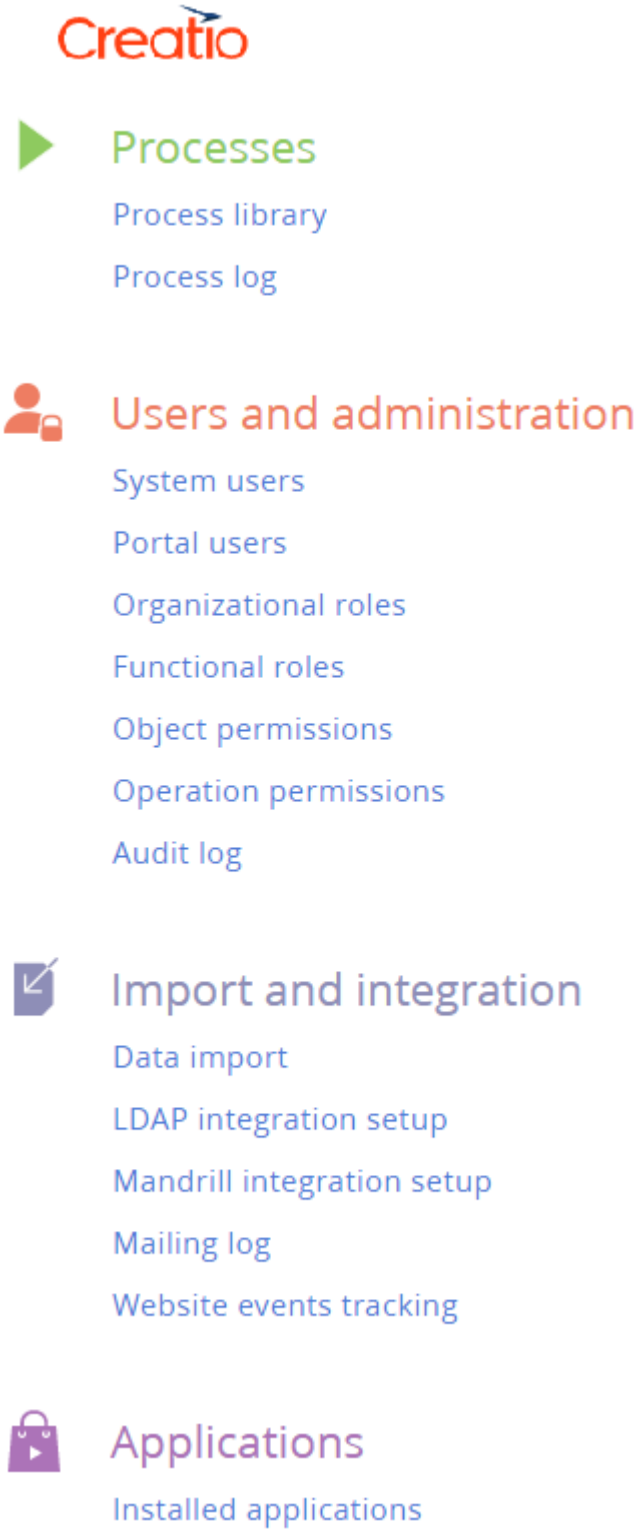
2. System behavior settings for the user:

- Notification settings.
- The default language settings that the user uses in the business card scanner.

4. Separate settings page

Putting all the settings on a separate page is the most convenient way for the user to manage the settings of the application. However, separate settings pages are also harder to implement (Fig. 1).

Fig. 1. Creatio settings page



You can open settings page from corresponding section and/or system designer.

It is recommended to put all application settings on a separate page in the following cases:

- If the settings should be carried out in a specific sequence.
- If a the setup requires both changes of settings and modification of lookups.
- If a the setup requires changes of settings, lookups and additional actions.
- If a user will need to update the settings during the work with the application on a regular basis.

Development according Creatio base logic

Correct filling of fields when creating leads

Creating leads when the Creatio is integrated with messengers, chats, landing pages or social networks is one of the typical integration tasks. When a lead is automatically created, it is necessary to keep correct logic of filling the fields of the lead page, including the information about lead generation channels.

When developing an application, make sure that Creatio users correlate each field name with its purpose and be able to use end-to-end analytics for the leads. An example of names of lead fields is given in the Table 3.

Table 2. Examples of lead page names

Name	Usage
[Registration method]	Lead registration method: • Automatically created • Added manually • Incoming call/email • Landing page • Case
[Channel]	Type of the resource from which the lead was received: • Web • Social networks • Offline advertising • Event • Recommendation/personal contact
[Source]	The name of the specific resource from which the lead was received (Twitter, Google, etc.).
[Transition website]	The site where the user clicked a link to the landing page, which resulted in creation of a new lead. The field is not editable and is filled in automatically when receiving a lead from the landing page.

UI minimum

Section icon

An icon should be added for each new section. The section icon requirements:

- PNG, or SVG image
- 38 x 38px
- A flat white color image without small elements/lines, without background.

You can find suitable icons in the Marketplace template library (Section icons for Creatio). You can also use the free service to search and convert flat icons.

Integration with Creatio and public API

Creatio has a wide range of integrations with custom third-party applications.

First, accessing Creatio by a third-party application requires authentication. For more information on accessing Creatio, see "Authenticating external requests to Creatio web services". For more information on Creatio primary authentication service, see "The AuthService.svc authentication service".

Starting with version 7.10, authentication is protected from CSRF attacks. For more information, see "Protection from CSRF attacks during integration with Creatio".

Brief description and comparison of Creatio basic integration methods is available in the "Choosing the method of integration with Creatio" article.

Wide range of integration capabilities is available through API provided by the DataService web service. Fore more information on create, read, update and delete operations (the CRUD operations), as well as the API, see the "DataService web service" article.

If the third-party system uses the OData protocol, it cal also be used for integration with Creatio. For more information, see the "OData" article.

Using the iframe HTML element for integration is covered in the "Integration of third-party sites via iframe" article.

For more information on the "Web-to-Object" integrations, see the "Web-To-Object. Using landings and web-forms article".

Marketplace application licensing

Introduction

Paid Marketplace applications must be licensed to monitor their use. The main license types are:

- Named licenses - grant application access for specific users. These licenses are linked to user profiles and can not be utilized by other users. The system administrator can redistribute registered licenses at any time. The process of distributing licenses among Creatio users is described in the License distribution article.
- Server licenses - do not provide for the need to license individual users. All Creatio users with the appropriate access rights will be able to access the licensed functionality.

Regardless of license type, it may include:

- Licensing objects – the names of custom objects added to the Marketplace application. This list can include any custom objects, such as a section, detail or lookup objects.
- Licensing operations – the names of operations added to the application logic to see whether a license is available for the use of certain functionality. For example, an additional action that requires the connection to the licensing operation has been added to one of the base system sections. When this action is triggered in the application, the license must be checked first, and based on that the functionality should either be continued or interrupted.

A Marketplace developer may learn about the amount of purchased and distributed user licenses by contacting Marketplace support (marketplace@creatio.com).

Licensing methods

Possible methods of licensing depend on the license type and the subject of licensing (table. 1).

Table 1. Possible licensing methods

License type	The subject of licensing	Business cases
Nominal	Objects	The Marketplace application is a new Creatio section. The developer would like to receive a fee for every user working in their section.
Nominal	Operations	The Marketplace application is a connector to a third-party service. The developer would like to receive a fee for every user with the access to the service. For example, a telephony connector with a single user subscription model.
Server-based	Objects	The Marketplace application is a new Creatio section. The developer would like to receive a fixed fee, regardless of how many users work with the section.
Server-based	Operations	The Marketplace application is a connector to a third-party service. The developer would like to receive a fixed fee, regardless of how many users work with the section. For example, a web-chat connector, used to register leads / cases in Creatio. The client purchases license regardless of the quantity of users.
Nominal	Objects and / or operations	The Marketplace application is a section with records that are created based on the data from third-party connectors to external services. The developer would like to receive a fee for every user with the access to the service. Additionally, the developer would like to receive a fee for every connector to the service. For example, the [Cases] section is modified in the vertical solution

framework. It integrates with the external system used to register cases. The client pays for access to the section functionality for each user.

ATTENTION

It is prohibited to use both nominal and server-based licenses that encompass the same objects and operations.

What is the validity period of a license

Every license includes the period of its validity. The Creatio platform monitors the validity of licenses. If the date has expired, the work of the licensed object or operation will be blocked.

Depending on the validity period, the licenses can be of two types:

- *Annual* - designed for Marketplace applications distributed with an annual subscription. The license is valid for one year. Upon expiration, the client must renew the annual subscription and get a new license.
- *Unlimited* – designed for Marketplace applications with a single payment method. The license is unlimited.

The process of issuing a license for a new product

1. When creating a Marketplace application, the developer must decide:

a) What will be included in the license - licensing objects or operations (both options are possible). They must also create a list of licensed objects and / or operations.

B) Determine the type of license - nominal or server-based. The main types of Creatio licenses and their usage options are listed in the "Creatio licensing" article.

c) Determine the license validity period – the license is annual or unlimited.

NOTE

If the license includes licensing operations, it is necessary to implement a license validation in the program code of the application. The verification logic for object licensing is already implemented in the base version of Creatio.

ATTENTION

The names of the licensed objects must contain the prefix specified the **developer profile settings**. Developer prefix – unique ID added to the names of custom schemas, packages and custom columns in the objects inherited from the base objects. The prefix identifies the functions added or modified by the developer. The prefix must be at least 3 characters long. The developer prefix cannot be changed after the registration.

2. Upon successful verification of the application by Marketplace employees, the developer must provide a list of licensed objects and operations, as well as the information about the type and duration of the license to the Marketplace support team (marketplace@creatio.com).

3. Based on the provided list, the Marketplace support service generates licenses for this application.

4. In the future, after ordering and paying for the Marketplace application, the user will have the necessary licenses installed by the Marketplace support service.

How to implement the license checking mechanism in a product

Checking licensing objects

If the license application includes only licensing objects, then no improvements in the Marketplace application are required. The license for licensing objects is checked by the base Creatio tools.

Checking licensing operations

If the application license includes a licensing operation, provide a license verification mechanism in the source code of the application to perform this operation.

To do this, use one of the methods of the *LicHelper* class of the *Terrasoft.Core* library:

- *GetHasOperationLicense* - returns *true* if the license is found, and *false* if the license is missing.
- *CheckHasOperationLicense* - returns *true* if the license is found. If there is no license, it generates the

LicException exception.

Method signatures:

```
public bool GetHasOperationLicense(string sysPackageOperationCode);
public bool CheckHasOperationLicense(string sysPackageOperationCode);
```

The *sysPackageOperationCode* string parameter must contain the name of the licensed operation passed to the Marketplace support service.

Examples of method use:

```
UserConnection.LicHelper.GetHasOperationLicense("MyMarketplaceApplication.Use");
UserConnection.LicHelper.CheckHasOperationLicense("MyMarketplaceApplication.Use");
```

ATTENTION

You can check the existence of an operation license only on the server side of the application. If the Marketplace application works only with the client's part of Creatio, you need to:

- Create a custom configuration service to implement a license validation;
- Add the program logic to the application to connect to the created service.

ATTENTION

We do not recommend verifying the existence of an operation license directly in custom schemas since the source code of a custom schema configuration packages is read/replace-only. Therefore, we recommend using licensing objects.

Application demo version

If the Creatio application does not have any downloaded licenses for any of the products, the application's demo mode is enabled by default. In demo mode, all Creatio functionality is available to the user, but not more than 1000 records may be added to each database table.

Application development examples

Developing functionality of a custom application is part of the general Marketplace application development sequence.

A detailed example of integration of a custom third-party service to a Creatio Marketplace application is covered in the article "**Developing a simple application for Creatio Marketplace**".

Developing an application, which would add a new section to Creatio is covered in the article "**Developing an advanced Marketplace application**".

The process of migrating demo data from a certain section and binding them to a package is covered in the "**Binding data to package**" article.

Developing a simple application for Creatio Marketplace

Introduction

Creatio Marketplace applications expand the features of the base Creatio products and provide additional business value. Developing an Creatio Marketplace application is similar to developing a customized Creatio configuration as part of an implementation project. The general principles of developing custom solutions for base Creatio products are covered in the "How to start development" article of the Development Guide.

Since Creatio Marketplace application are a form of customizing base Creatio products, they must somehow be deployed on existing Creatio applications. The deployment mechanism for Marketplace applications is implemented via packages. A Creatio package is a complex of configuration elements, that implement a set of functions.

The number of packages and their contents depend on the scope and complexity of functions that the packages implement. For example, calling a third-party service from an exiting record edit page requires replacing a couple of schemas that can be grouped in a single package. On the other hand, creating a new Creatio section with complex functions requires developing several dozens of schemas and modules.

The procedure for developing a simple Creatio Marketplace application is as follows:

1. Create a Web-service, if needed. A Web-service can be implemented via Creatio configuration elements or as a third-party solution.

2. Create a custom package.

3. Add a replacing view model schema for main menu.

4. Implement the necessary new functions in the replacing schema.

Application details

Create an application that would call a third-party Web-service that would display a greeting message on the main menu page of Sales Creatio, enterprise edition. Web-service calling is implemented via an HTTP POST request, that passes current user name as parameter. The result must be displayed under the social network links.

Implementation algorithm

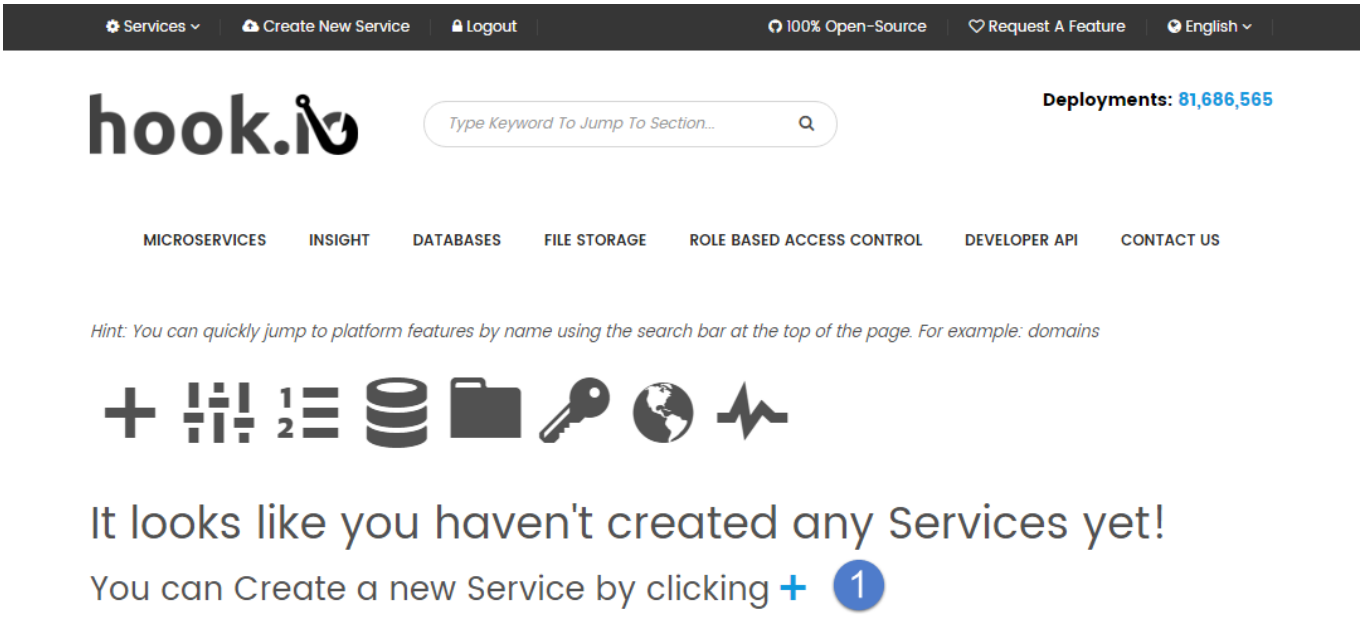
1. Web-service implementation

There are a number of ways (different programming languages and platforms) to implement a Web-service. The best option depends on the purpose of the Web-service.

In this case, a simple "microservice" will be enough. Use any of the available platforms, such as open-source project https://hook.io to implement this service.

After you register and log in to the system, a miscoservice management page will open where you can add a new Web-service (Fig. 1, 1).

Fig. 1. hook.io microservice management page



Add a new service, specify its endpoint name (Fig. 2), select programming language for the source code of the service (Fig 3, 1) and add service source code (Fig 3, 2).

Fig. 2. Adding endpoint

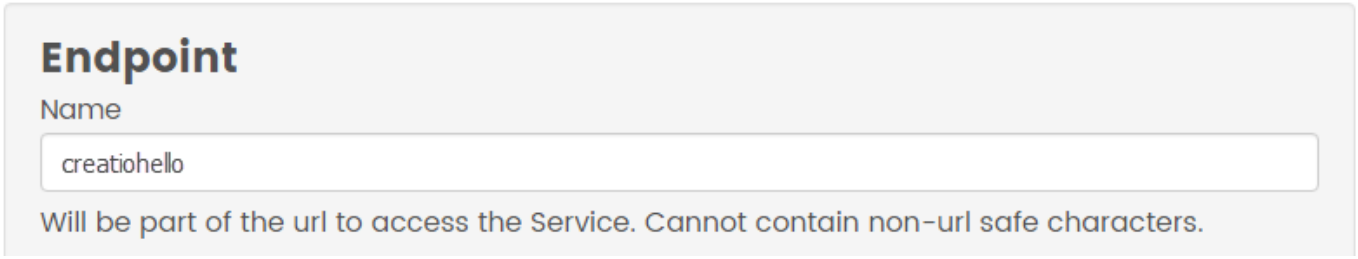
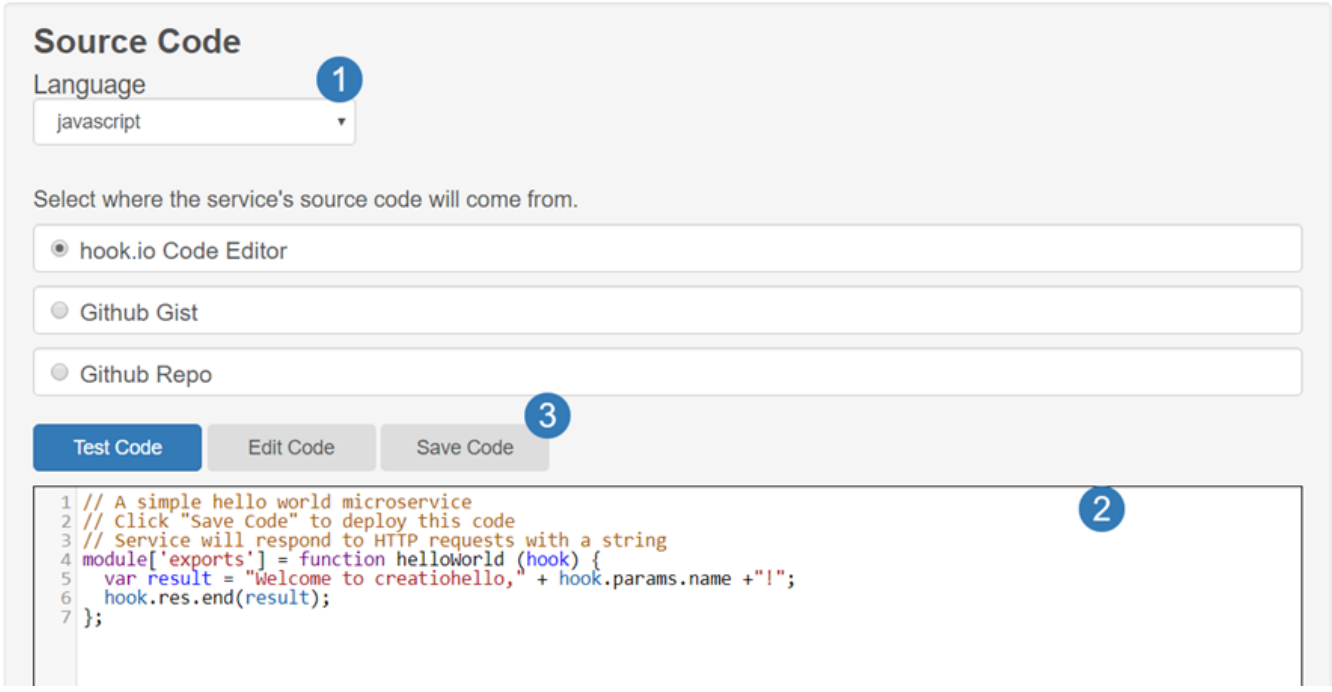


Fig. 3. Adding microservice source code



After saving the source code (Fig 3, 3) Endpoint address is https://hook.io/academy-creatio-com/creatiohello.

Service source code is a function that generates the greeting message, based on the *name* parameter of the HTTP request. Microservice source code:

```
module["exports"] = function helloWorld (hook) {
  // Generating a reply string.
  var result = "Welcome to creatiohello, " + hook.params.name + "!";
  // Sending responde to the client.
  hook.res.end(result);
};
```

NOTE

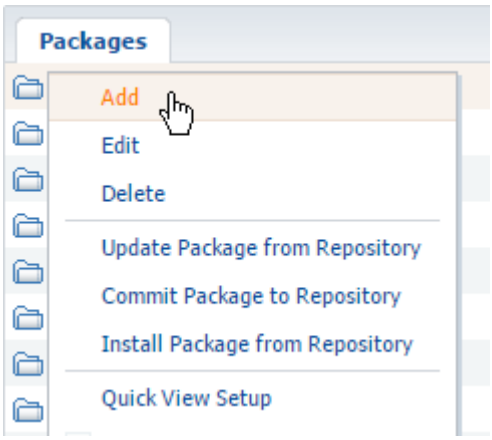
This source code works with both POST and GET methods of the received HTTP requests. For example, if you enter https://hook.io/academy-creatio-com/creatiohello?name=User in the browser address bar, the service will still process it correctly.

2. Creating a custom package

Creating and editing custom packages is done in the [Configuration] section.

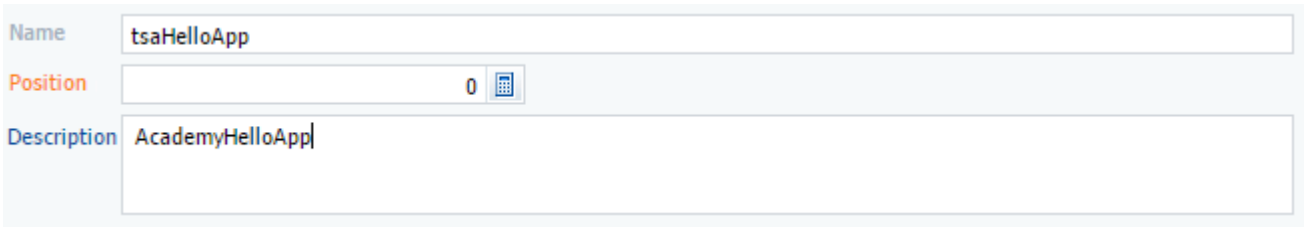
To create a new custom package, go to the [Configuration] tab of the [Advanced settings] window, right-click the [Packages] panel select and select [Add] (Fig 4).

Fig. 4. Adding a custom package



As a result, a package edit page will pop up, where you can fill out the properties of the new package (Fig. 5).

Fig. 5. Basic properties of a package



Package properties include:

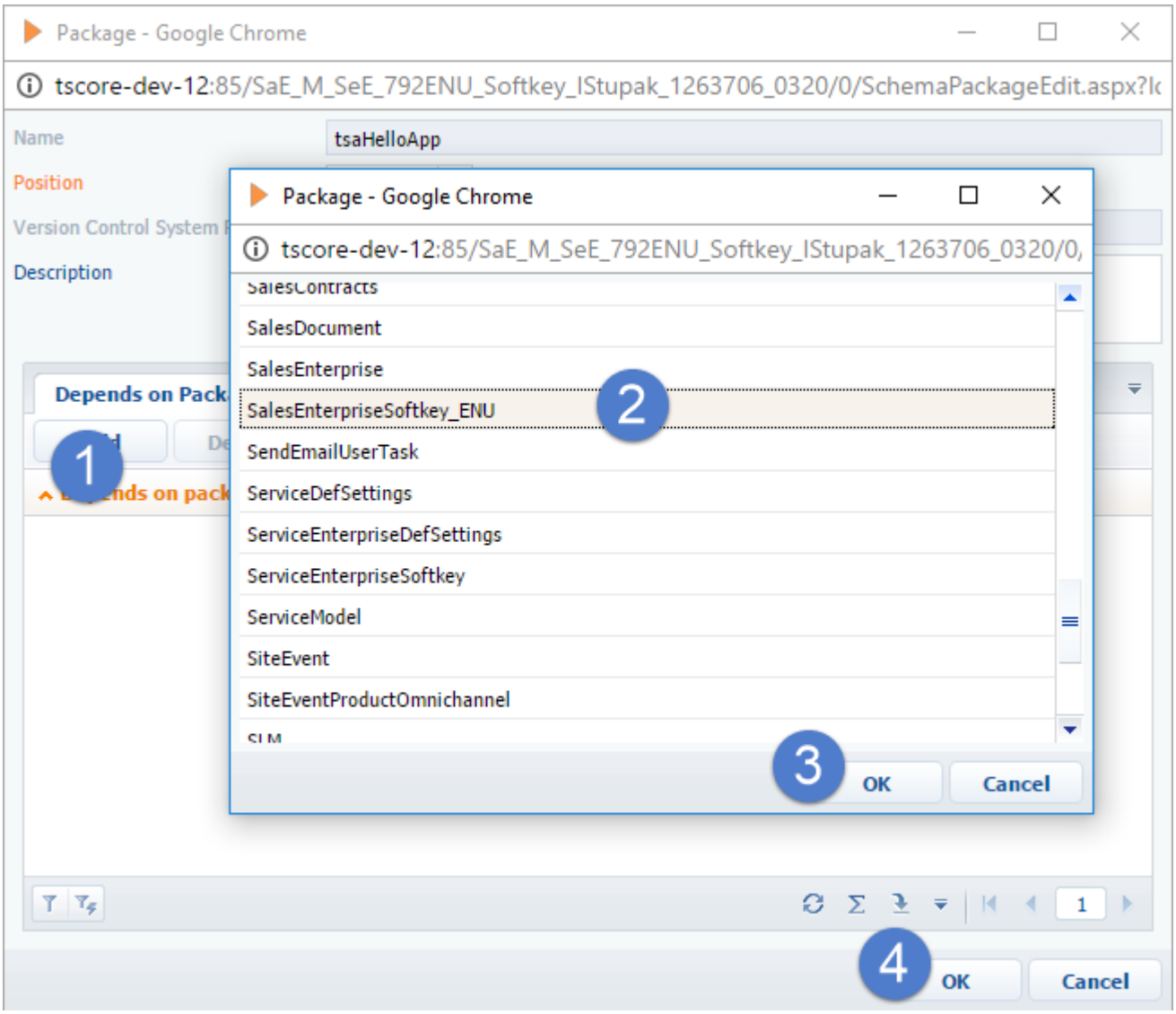
- [Name] – package name. This is a required field. Package name cannot match other package names. The name must include developer's prefix specified in the developer profile.
- [Description] – short package description, such as information about implemented functions. This field is optional.

Click [OK] to save the new package. The package will become available in the list of other packages on the [Packages] tab.

Set up dependencies for the new package to ensure that it includes existing functions. Specify the last element in the package hierarchy. Use the [Package Dependencies] tab in the [Configuration] section to determine the last package in the hierarchy. Find the first package located above the [Custom] package.

To add a dependency, click the [Add] button on the [Depends on packages] tab of the package edit page (Fig 6, 1). In the package lookup, select the needed package (Fig 6, 2) and click [OK] (Fig. 6, 3). Save the package after adding all dependencies (Fig. 6, 4).

Fig. 6. Adding package dependencies



3. Creating a replacing schema

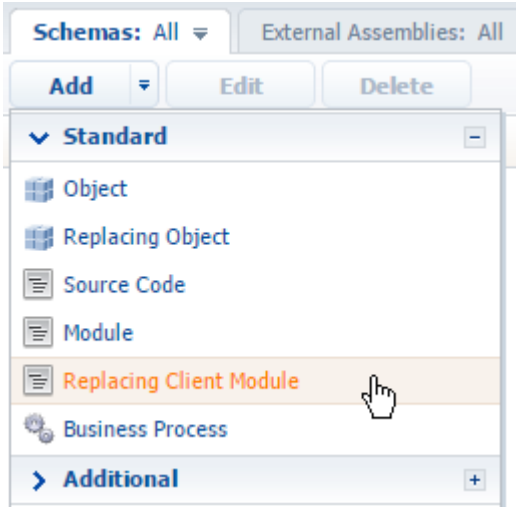
Creatio main menu is based on several view model schemas that inherit each other hierarchically:

- The base schema of the main page (*BaseIntroPageSchema*, in the package *UIv2*), which is the parent view model schema where the primary containers of the main page are generated. Adds a left/right separator to the view model, as well as video and social network link panels.
- Base product main menu schema (*SimpleIntro*, in the package *UIv2*), which is inherited from the *BaseIntroPageSchema*. Adds the [General], [Analytics] and [Admin area] blocks to the main menu.
- Sales enterprise main menu schema (*EnterpriseIntro*, in the package SalesEnterprise), which is inherited from the *SimpleIntro*. Adds more [General] section links, as well as the [Sales] section. It also redefines the link to the overview video.

To make custom changes in the main menu of the sales enterprise product, replace the *EnterpriseIntro* schema and implement additional functions in the replacing schema.

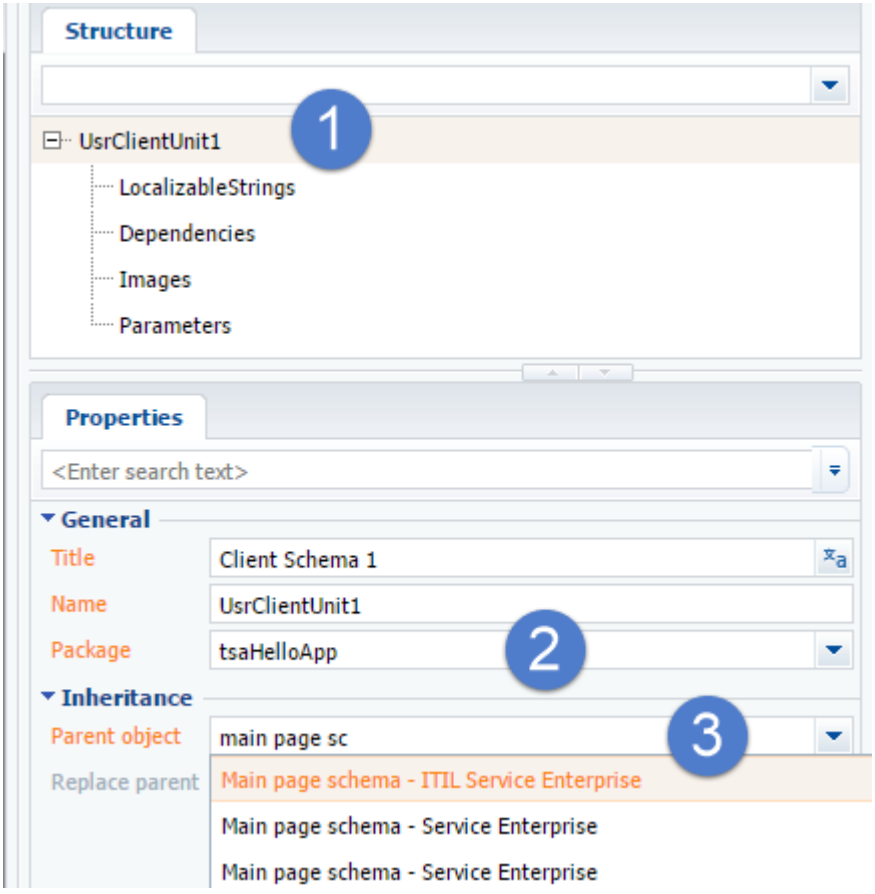
To create a replacing schema, select the package that you added earlier on the [Packages] panel of the [Configuration] tab of the advanced settings window. On the [Schemas] tab, select [Add] > [Replacing client module] (Fig. 7).

Fig. 7. Adding replacing client module



In the opened client schema designer window, open the root element in the [Structure] area (Fig 8, 1), then select the replacing schema in the [Parent object] field. You can start typing the title of the replacing schema in the [Parent object] field to filter the list of available schemas (Fig. 8, 3). Select “Main page schema - Sales Enterprise”.

Fig. 8. Selecting parent schema



As a result, a replacing schema for the main page view model will be added to the custom package.

4. Implementing functions

To implement custom functions in the new schema, add its source code.

To display the result of calling a custom microservice on the main page, add configuration objects for the container and the text in it to the *diff* array of the view model schema.

Develop a method for calling the *SetHelloAttribute(),* micriservice, in which the *XMLHttpRequest* JavaScript object will be used. This method must be called on page loading. To do this, override the *Init()* method of the parent schema and call *SetHelloAttribute()* in it. The result of executing the request to the microservice can be obtained from the callback unction of the *onreadystatechange* property of the *XMLHttpRequest* object. Since the request is executed asynchronously, the result must be saved in the *HelloAttribute* attribute. The attribute must be connected to the caption property of the text that will display the message.

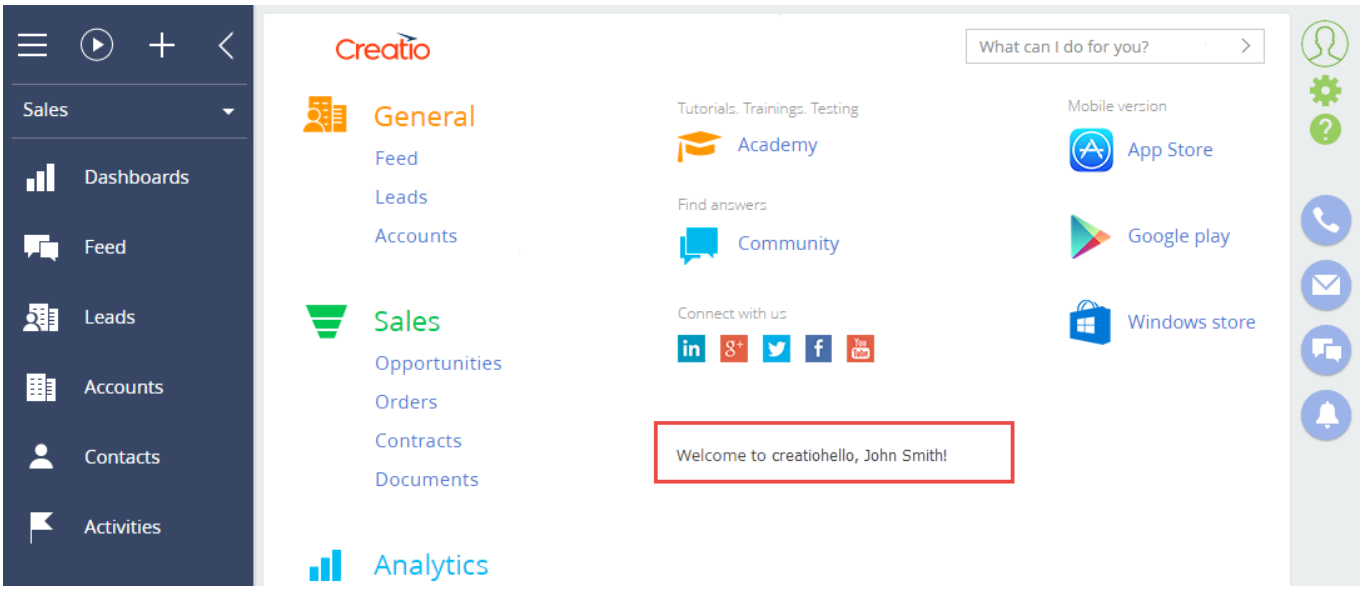
Source code of the main page view model:

```
define("EnterpriseIntro", [], function() {
    return {
        attributes: {
            // Attribute that contains the message from the custom Web-service.
            "HelloAttribute": {
                // Data type of the attribute.
                "dataValueType": Terrasoft.DataValueType.TEXT,
                // Attribute type - virtual column.
                "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
                // Default value.
                "value": ""
            }
        },
        methods: {
            // Schema initialization method.
            init: function() {
                // Calling base functions.
                this.callParent(arguments);
                // calling custom service.
                this.SetHelloAttribute();
            },
            // Custom service call method.
            SetHelloAttribute: function() {
                // HTTP request.
                var xhr = new XMLHttpRequest();
                // Custom service call URL.
                var url = "https://hook.io/academy-creatio-com/creatiohello";
                // Determining current user name.
                var currUser = Terrasoft.SysValue.CURRENT_USER_CONTACT.displayValue;
                // Generating parameters for HTTP request.
                var params = "name=" + currUser;
                xhr.open("POST", url, true);
                xhr.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
                // Saving context.
                var self = this;
                // Callback function that is called upon receiving reply.
                xhr.onreadystatechange = function() {
                    // If a reply with the needed status is received.
                    if (xhr.readyState === 4 && xhr.status === 200) {
                        // Setting attribute value.
                        self.set("HelloAttribute", xhr.responseText);
                    }
                    else {
                        self.set("HelloAttribute", "creatiohello is unavailable");
                    }
                };
                // Sending HTTP request.
                xhr.send(params);
            }
        },
        diff: [
            // Custom container, which will contain the greeting.
```

```
            {
                // Insert operation.
                "operation": "insert",
                // Element name.
                "name": "HelloContainer",
                // Name of the parent container element.
                "parentName": "RightContainer",
                // Parent element property name for embedded elements.
                "propertyName": "items",
                // Element properties configuration object.
                "values": {
                    // Element type - container.
                    "itemType": Terrasoft.ViewItemType.CONTAINER,
                    // Array of embedded elements.
                    "items": []
                }
            },
            // Greeting text label.
            {
                "operation": "insert",
                "name": "HelloLabel",
                "parentName": "HelloContainer",
                "propertyName": "items",
                "values": {
                    // Element type - label.
                    "itemType": this.Terrasoft.ViewItemType.LABEL,
                    // Link to an attribute with the label text.
                    "caption": { "bindTo": "HelloAttribute" }
                }
            }
        ]
    };
});
```

After saving the schema, update the main page in the browser. The result of the creatiohello microservice request will be displayed on it.

Fig. 9. Case result



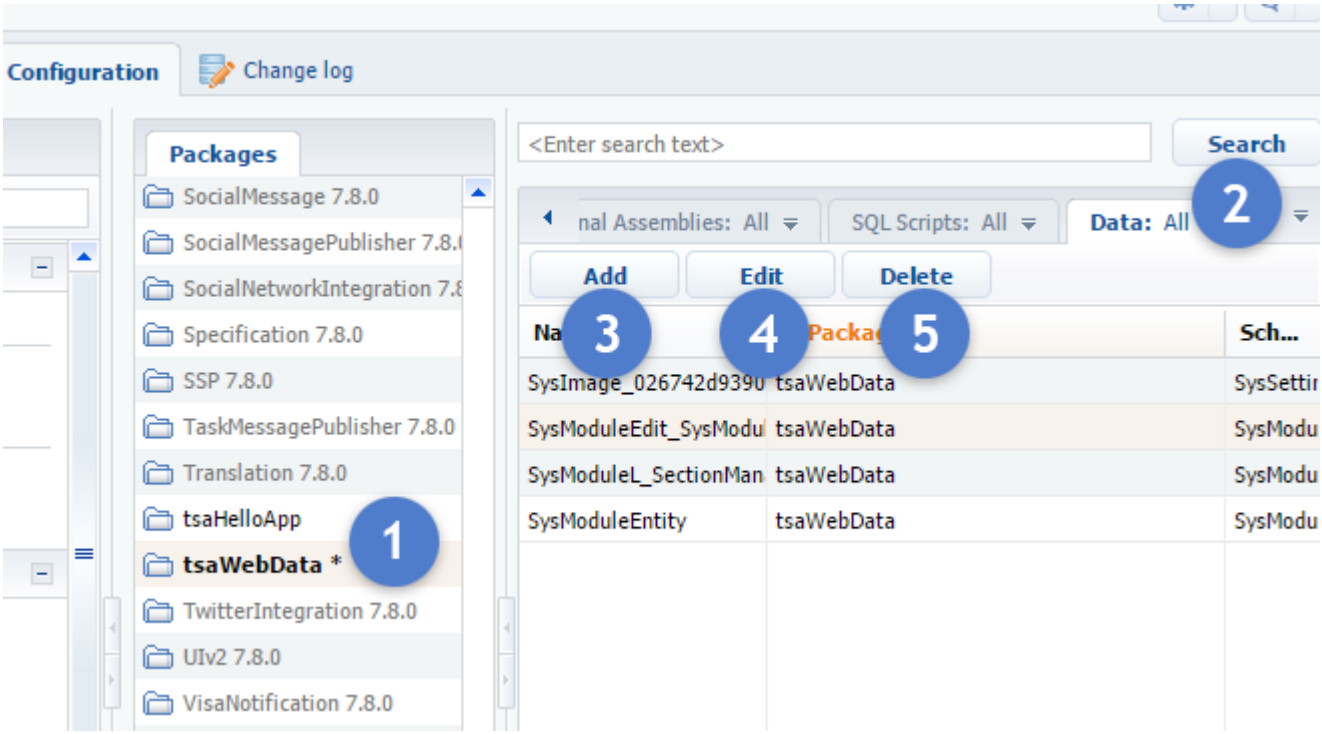
Binding data to package

Introduction

Once the Section Wizard automatically attaches the data of a custom section to the custom package. However, for the section to show up in a workplace and with demo records, you need to attach additional data.

The data is attached in the [Configuration] section. To view the data attached to the package, select the necessary package (Fig. 1,1) and select the [Data] tab (2). You can add (3), edit (4) and delete (5) the data bound to package.

Fig. 1. The [Data] tab of the [Configuration] section

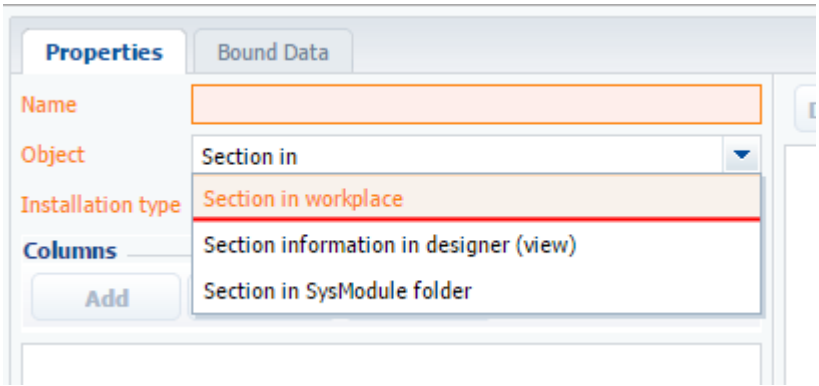


The following data attachment examples are based on the standard approach to creating user sections, described in the "**Developing an advanced Marketplace application**" article.

Attaching sections to a workplace

Bind the data of the [Section in workplace] for the section to show up in the [Marketing] workplace. Click [Add], and select the "Section in the workplace" value in the [Object] field of the package data attachment window (Fig 2).

Fig. 2. Selecting an object for data attachment

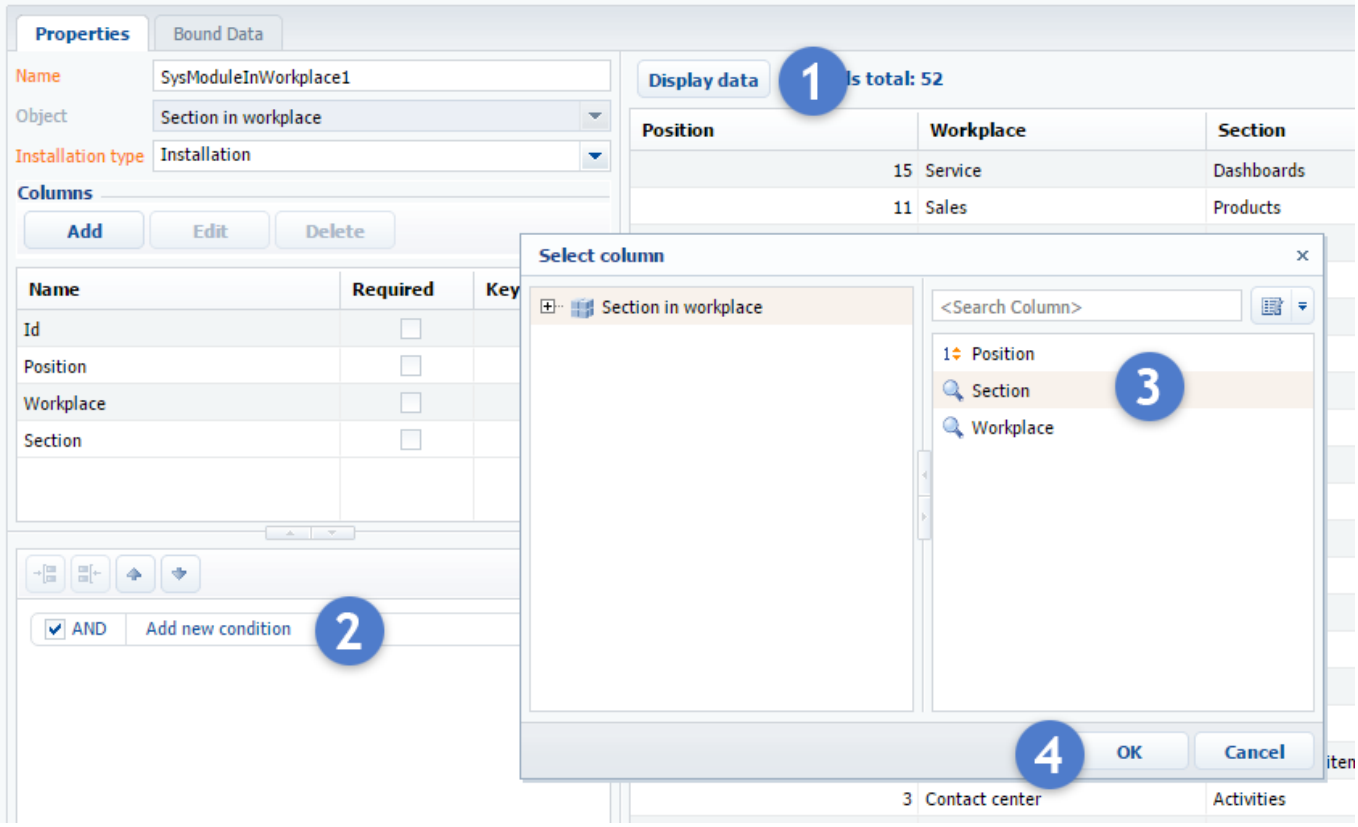


Click [Yes] in the pop-up window.

Select the installation type. The universal option is "Installation". If you select "Installation", the data will be attached after the package installation and with subsequent updates.

Click [Display data] (Fig. 3, 1), to view information about all sections and workplaces currently in development.

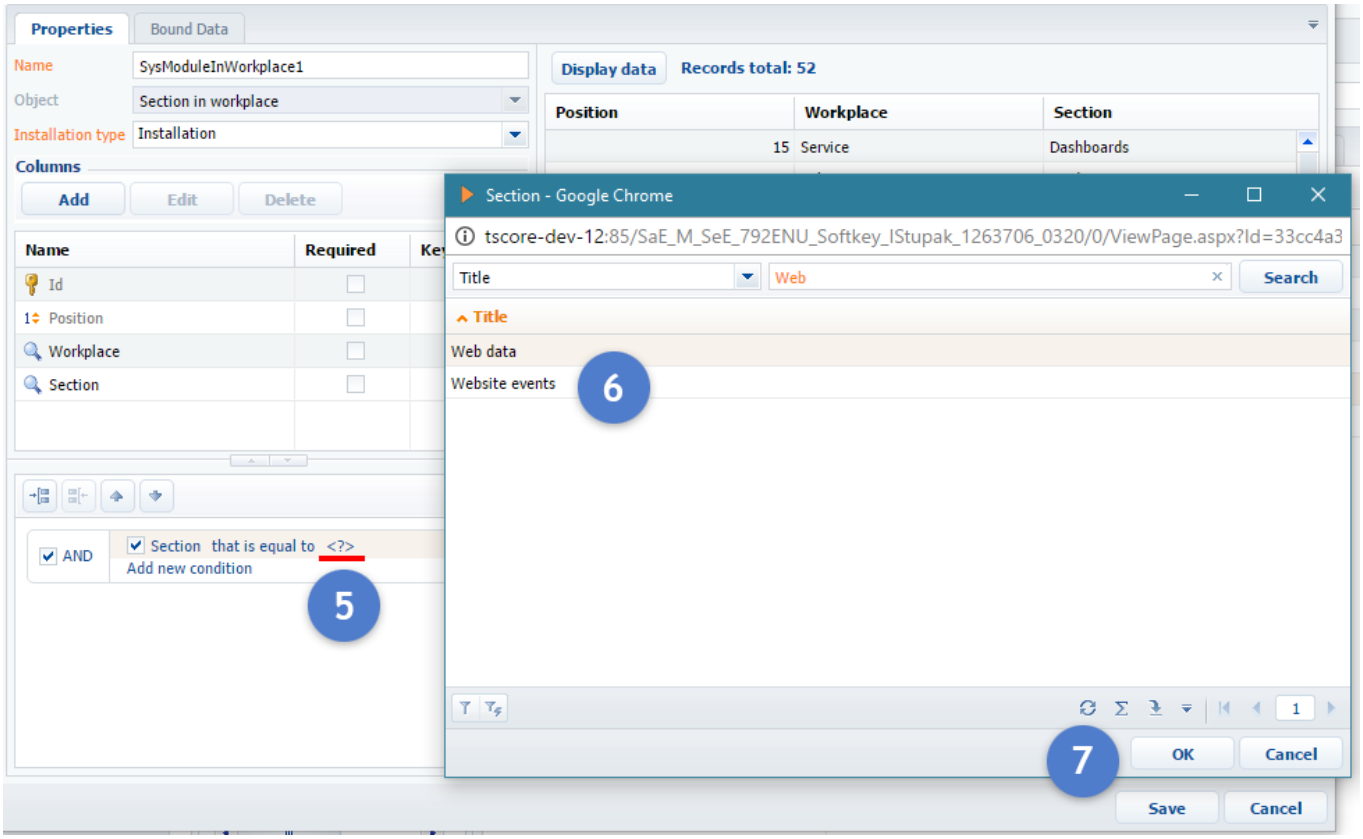
Fig. 3. Displaying and filtering attached data



Prior to attaching the data to the created section, you need apply a filter. In the filter area, add a new condition (Fig. 3, 2), select a column to filter by (3) and click [OK] (4).

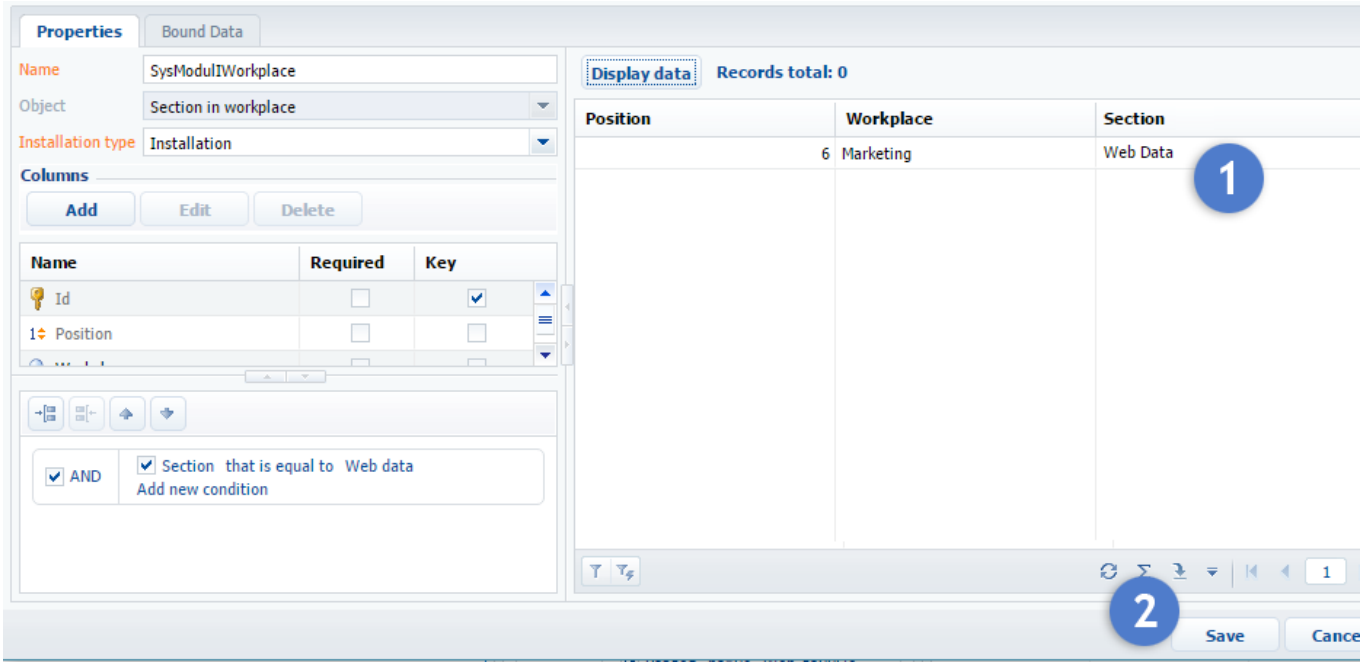
Build the second part of the condition. Click on the condition sign (Fig. 4, 5), select the name of the section (6), and click [OK] (7).

Fig. 4 Constructing a data filtering condition



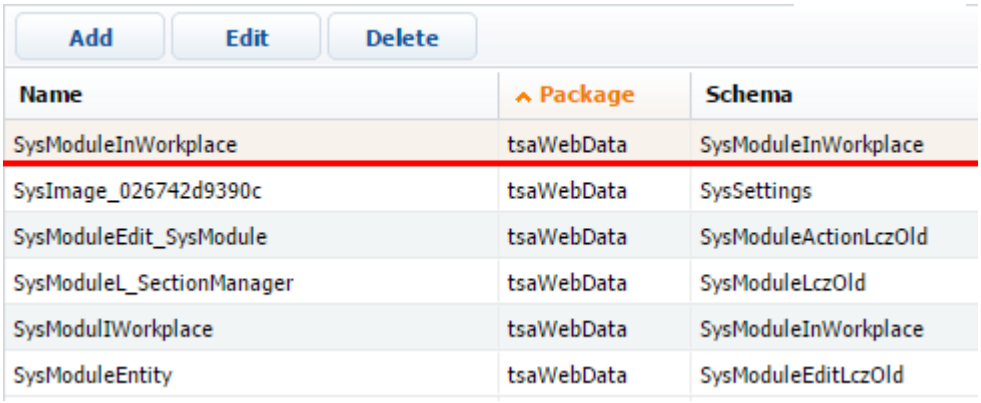
The filter for selecting the created section is built as the result (Fig. 5, 1).

Fig. 5. Fig. 4 Constructing a data filtering condition



A new entry will show up in the [Configuration] section of the [Data] tab (Fig. 6) after saving the attached data (Fig. 5, 2).

Fig. 6. Attached data of the [Section in workplace] schema



Section data attachment

To attach the demo content of the section (see Fig. 7), you must first determine the title of the main section object schema (see Fig. 8).

Fig. 7. Section demo content

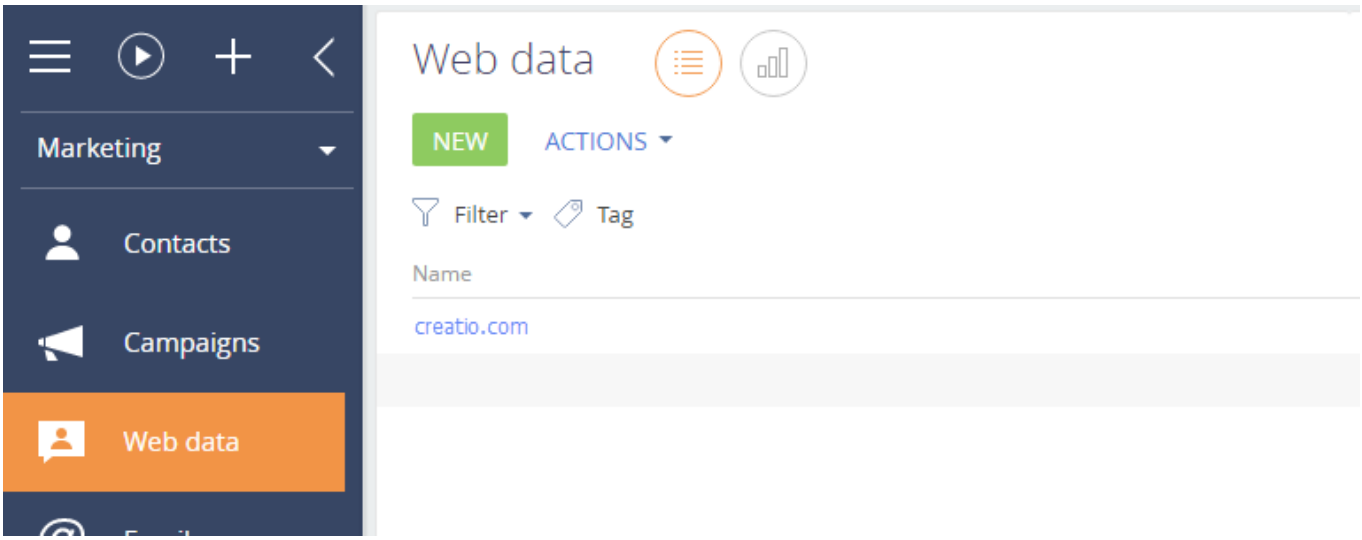
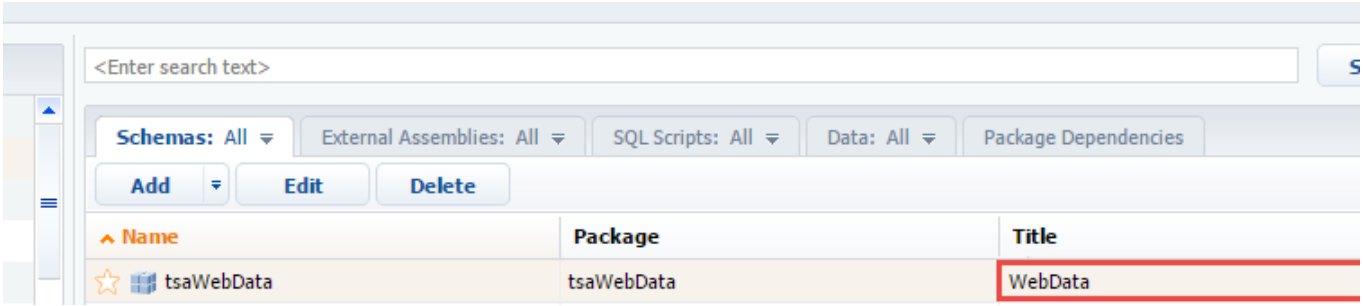


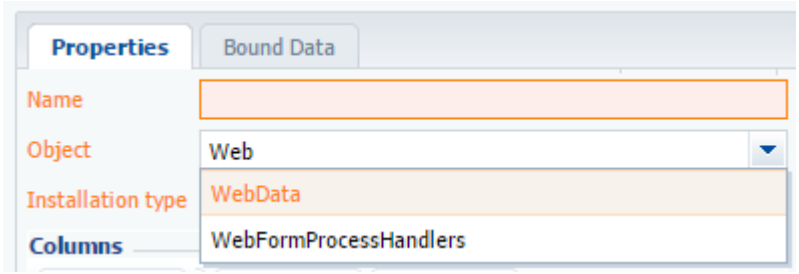
Fig. 8. Section object schema header



This schema is created automatically by the Section Wizard and the title of the schema should match the section header.

Click [Add], and select the "Section in the workplace" value in the [Object] field of the package data attachment window (Fig 9).

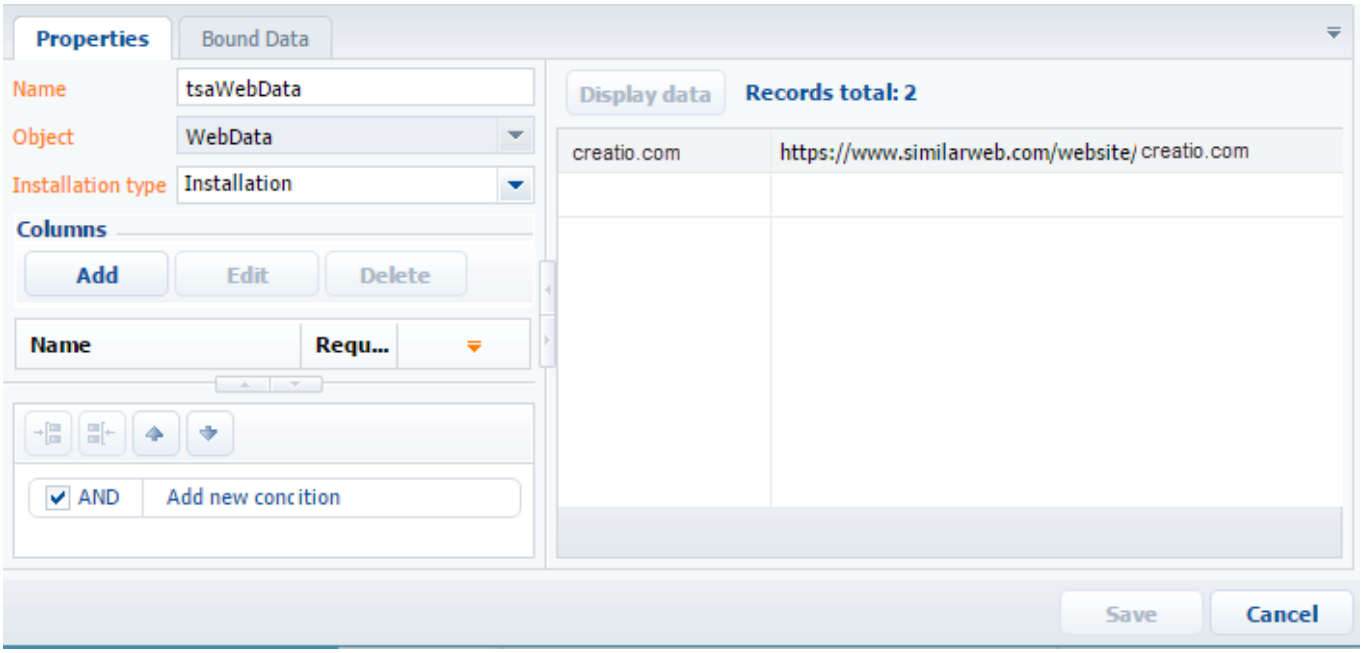
Fig. 9. Selecting an object for data attachment



Click [Yes] in the pop-up window.

Select the installation type “Installation”. Clicking the [Display data] button to view the attached data (Fig. 10). Use filter to select specific records.

Fig. 10. Connected demo data



Click the [Save] button to save and close the data attachment window.

Developing an advanced Marketplace application

Introduction

Developing a Creatio Marketplace app with a custom section is identical to developing a project solution with a custom section. Please refer to "Section business logic" for more insights about the main principles of custom section development.

Use the Section Wizard to create a new custom section. All schemas created by the Section Wizard are stored in the package specified in the [Current Package] (*CurrentPackageId*) system setting. By default, this is the [Custom] package.

Since [Custom] is a system package and cannot be exported using SVN or WorkspaceConsole, you cannot use this package for development purposes (if the changes must be transfered to other environments).

NOTE

The schema export and import mechanism will enable you to export the schemas from [Custom] package. However, you cannot transfer the package data and SQL scripts using this mechanism.

Specify a custom package in the [Package] property of a custom schema to move the schema to that package upon saving.

Therefore, all development must be carried out in custom packages.

To create an app with a custom section:

1. Create a custom package.

2. Set values of the [Current package] and [Object Name Prefix] system settings to begin development in the custom package. The value of the [Object Name Prefix] system setting should match the prefix specified in the Marketplace Developer profile (See: "**Developer profile setup**").

3. Create a new section using the Section Wizard.

4. Implement the necessary features.

5. Connect the necessary section data to your custom package.

Application details

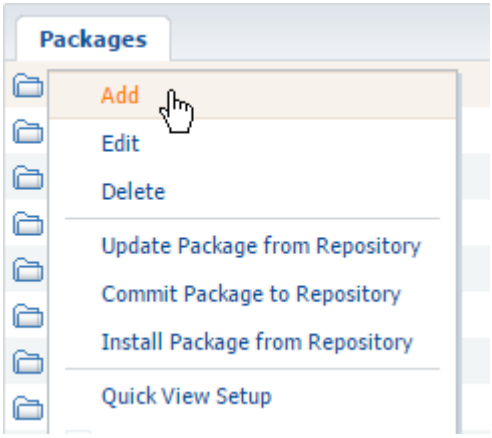
Create a new section called [Web data] in the [Marketing] workplace. Implement displaying of the web stats based on the website URL on the section page. Display statistics in an <iframe> element using a third-party website https://www.similarweb.com.

Implementation algorithm

1. Creating a custom package

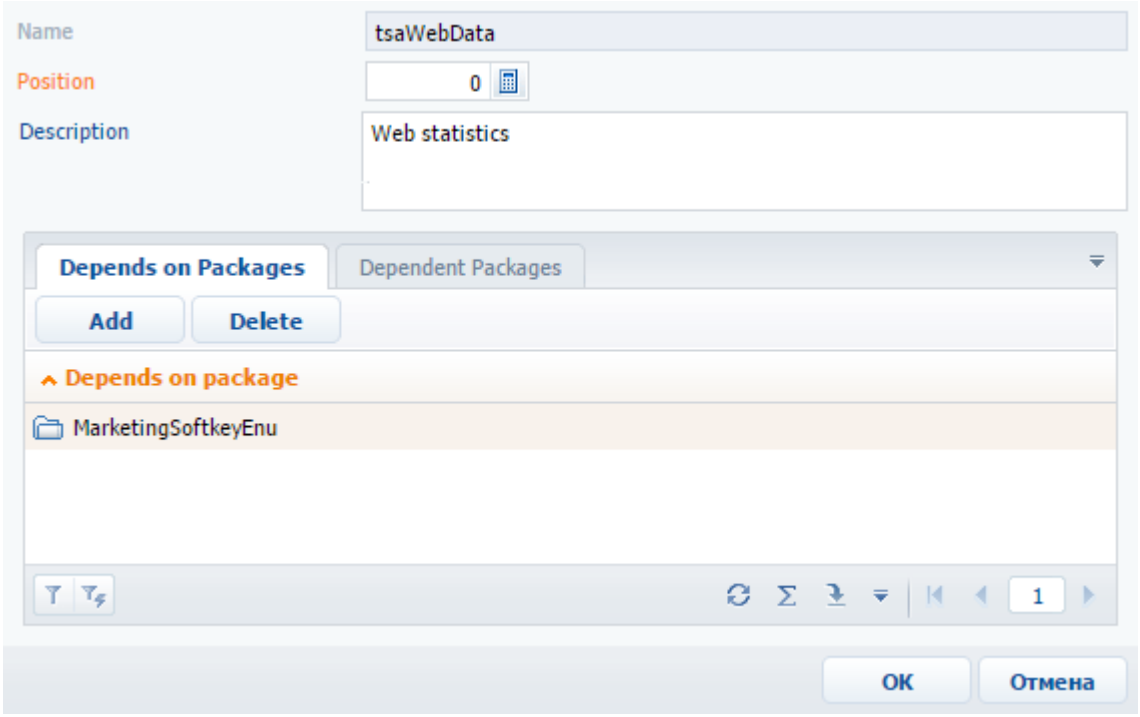
To create a new custom package, go to the [Configuration] tab of the [Advanced settings] window, right-click the [Packages] area and select [Add] (Fig 1).

Fig. 1. Adding a new package



Fill out the primary field values in the package page and save it. Add package dependencies by opening the page again. Since the section must be displayed in [Marketing] workplace, add a dependency from the [MarketingSoftkey] package (Fig. 2).

Fig. 2. Package page



ATTENTION

The package name should include the prefix that you specified in the Developer profile (See: “**Developer profile setup**”).

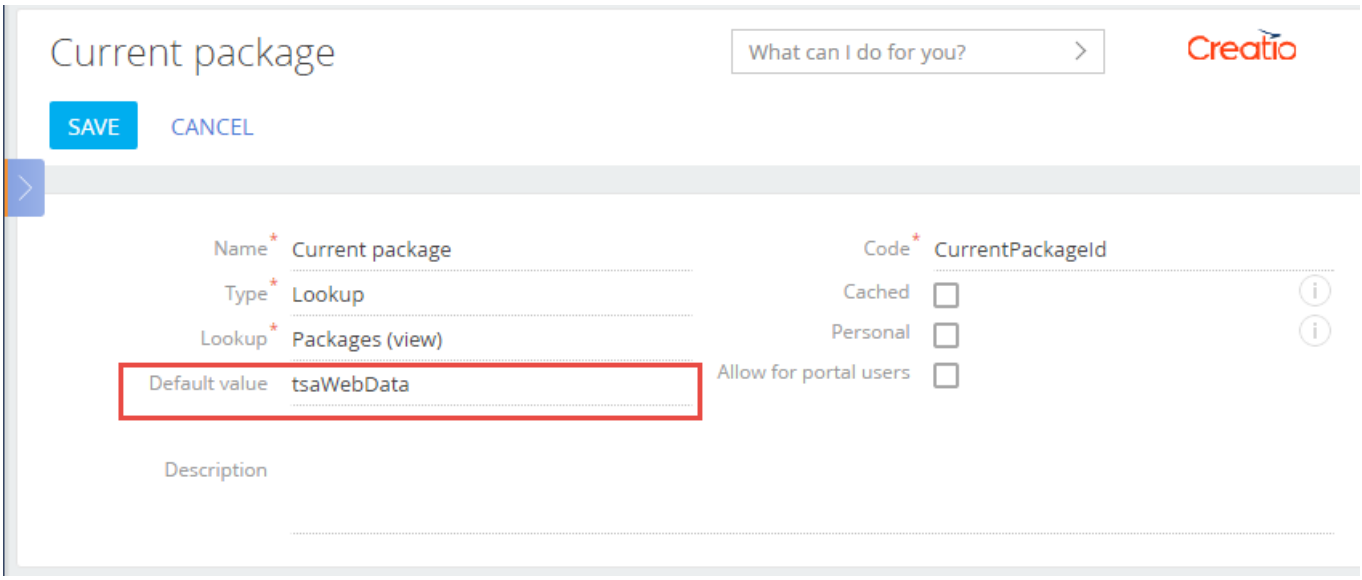
NOTE

The specified dependencies of the package will influence the compatibility between your application and other Creatio products.

2. Setting the values of the [Current package] and [Object name prefix] system settings

To save section schemas generated by the Section Wizard in your custom package, specify it as the “current package”. Go to the [System settings] section, open the [Current package] system setting and select the needed custom package in the [Default value] field (Fig. 3).

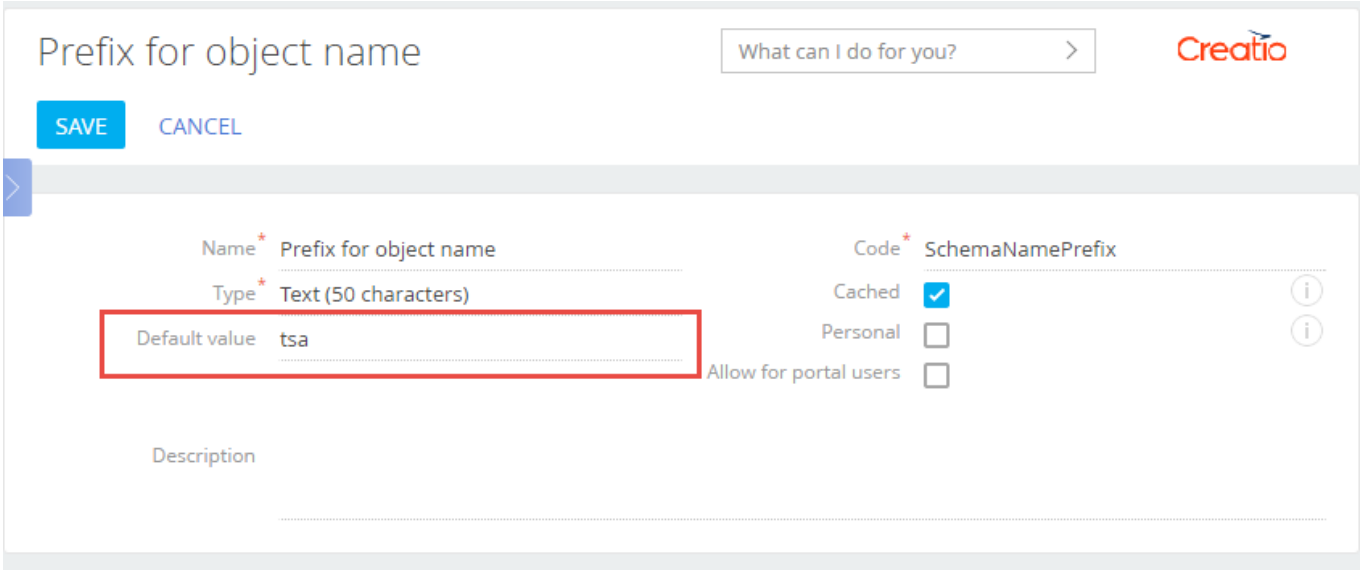
Fig. 3. Editing [Current package] system setting



Save the system settings.

Specify the prefix value from the **Developer profile** in the [Object name prefix] system setting (Fig. 4).

Fig. 4. Filling in the [Object name prefix] system setting



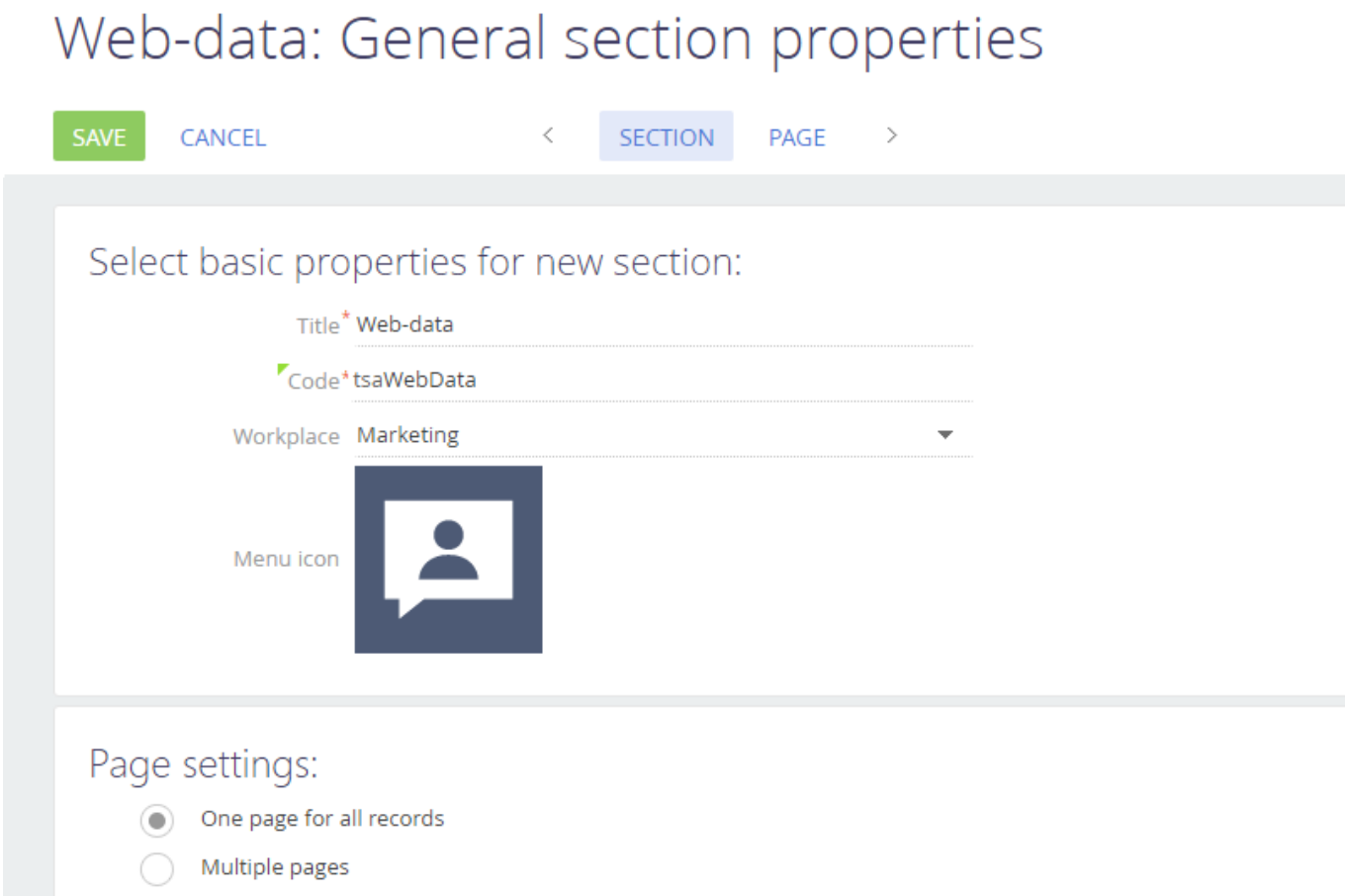
This prefix will be automatically added to the names of the schemas created by the Wizard.

3. Using the Section Wizard to create sections

Please refer to "Section Wizard", "How to configure section properties", and "How to configure section page" articles for more information about sections and the Section Wizard.

Fill out the values for the following properties (Fig. 5):

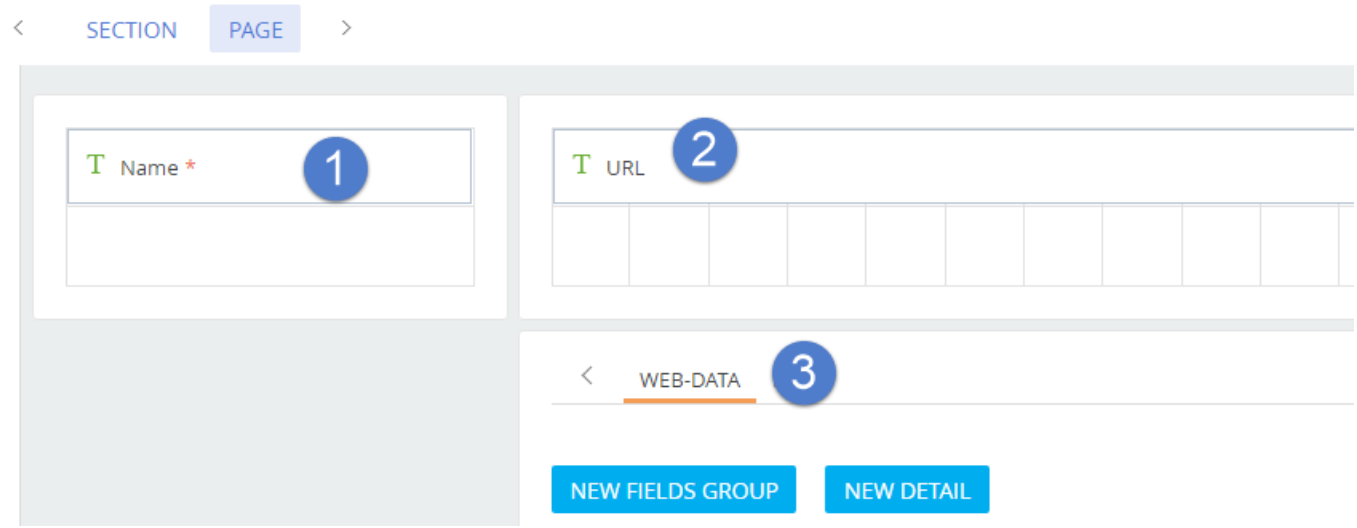
Fig. 5. Section properties in the Section Wizard



- [Title] – “Web-data”. The title is displayed in the main menu and on the section page.
- [Code] – "tsaWebData". The name of the schema of the object in the section (where “tsa” is the prefix that was added automatically). Find out about the schemas created by the Section Wizard in “How to configure section properties”.
- [Workplace] – “Marketing”. Your section will be available in this workplace.

The section edit page must have a field for entering website URL. You can use the default [Name] field, which is automatically created by the Wizard (Fig. 6, 1) for this purpose. Add an extra field (2), which will contain the link to the website stats from https://www.similarweb.com for the specified URL. This link will be assigned to the *src* attribute of the <iframe> element of the page. To place the <iframe> element, add the [Web-data] tab to the tab panel (3).

Fig. 6. Adding fields and tabs to the records edit page



Click the [Save] button. The Section Wizard will create the necessary schemas in the corresponding package and save the results in the database.

NOTE

It might take a few minutes for the Wizard to finish up the processes. You will be notified once the new section is saved.

Restart the browser and clear your cache to view the new section.

4. Implementing the section features

If you created your section in the wizard, the base features for adding, deleting, and sorting records will automatically be available in the section. The features for saving and loading the record data are automatically inherited in the schema of the edit page as well.

You need to create a button to generate the link to the stats page for the website specified in the URL field. To add a button, use the configuration object and add it to the *diff array*. The caption of the button is connected to the localized string *AddUrlButtonCaption*. The *click* event must be connected to the *addUrl() method*, which is used to form the link to the stats page and to set the new value for the *tsaURL*, associated with the [URL] field of the edit page. This link is also assigned to the *src* attribute of the <iframe> element. Learn more about adding buttons on the editing page in the “Adding a button” article.

The <iframe> element is added to the tab through a configuration object, which is added to the *diff* array. The <iframe> html element with the specified styles will be generated in the html properties of the configuration object. At the same time the *src* attribute is not indicated, since it’s generated in the *addUrl()* method. To restore the data in <iframe> after switching to another tab, connect the *afterrerender* event to the method *addUrl()*.

Upon opening the saved record, the *afterrerender* event usually occurs before the data is downloaded. This is why the *addUrl()* method should also be called in the method-handler *onEntityInitialized()*.

The source code of the view model schema of the edit page:

```
define("tsaWebData1Page", [], function() {
```

```
return {
    entitySchemaName: "tsaWebData",
    details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
    diff: /**SCHEMA_DIFF*/[
        // Name of the Web-page.
        {
            "operation": "insert",
            "name": "tsaName",
            "values": {
                "layout": {
                    "colSpan": 24,
                    "rowSpan": 1,
                    "column": 0,
                    "row": 0,
                    "layoutName": "ProfileContainer"
                },
                "bindTo": "tsaName"
            },
            "parentName": "ProfileContainer",
            "propertyName": "items",
            "index": 0
        },
        // Link to Web-page statistics.
        {
            "operation": "insert",
            "name": "tsaURL",
            "values": {
                "layout": {
                    "colSpan": 24,
                    "rowSpan": 1,
                    "column": 0,
                    "row": 0,
                    "layoutName": "Header"
                },
                "labelConfig": {},
                "enabled": true,
                "readonly": true,
                "bindTo": "tsaURL"
            },
            "parentName": "Header",
            "propertyName": "items",
            "index": 0
        },
        // Tab on the tab panel.
        {
            "operation": "insert",
            "name": "TabData",
            "values": {
                "caption": "Web-data",
                "items": []
            },
            "parentName": "Tabs",
            "propertyName": "tabs",
            "index": 0
        },
        // The [NEW URL] button.
        {
            "operation": "insert",
            "parentName": "ProfileContainer",
            "propertyName": "items",
            "name": "AddUrlButton",
            "values": {
```

```
                    "layout": {
                        "colSpan": 24,
                        "rowSpan": 1,
                        "column": 0,
                        "row": 1
                    },
                    "itemType": Terrasoft.ViewItemType.BUTTON,
                    "caption": {"bindTo": "Resources.Strings.AddUrlButtonCaption"},
                    "click": {"bindTo": "addUrl"},
                    "style": Terrasoft.controls.ButtonEnums.style.BLUE
                }
            },
            // Container with an embedded iframe HTML element.
            {
                "operation": "insert",
                "name": "IFrameStat",
                "parentName": "TabData",
                "propertyName": "items",
                "values": {
                    "id": "testiframe",
                    "itemType": Terrasoft.ViewItemType.CONTAINER,
                    "selectors": {"wrapEl": "#stat-iframe"},
                    "layout": { "colSpan": 24, "rowSpan": 1, "column": 0, "row": 0 },
                    "html": "<iframe id='stat-iframe' class='stat-iframe'
width='100%' height='550px'" +
                        "style = 'border: 1px solid silver;'></iframe>",
                    "afterrerender": {
                        "bindTo": "addUrl"
                    }
                }
            }
        ]/**SCHEMA_DIFF*/,
        methods: {
            // Hangdler for data load completion event.
            onEntityInitialized: function() {
                // Calling parent implementation of method.
                this.callParent(arguments);
                // Calling a method to add URL to iframe Html element
                this.addUrl();
            },
            // Method for adding URL to the iframe Html element.
            addUrl: function() {
                // Getting component by id.
                var iframe = Ext.get("stat-iframe");
                if (!iframe) {
                    window.console.error("A tab with the iframe is not found");
                    return;
                }
                // Getting the value in the [tsaName] column.
                var siteName = this.get("tsaName");
                if (!siteName) {
                    window.console.error("Website name not specified");
                    return;
                }
                // Generating a link to the statistics page.
                var url = "https://www.similarweb.com/website/" + siteName;
                // Setting the value of the [tsaName] column.
                this.set("tsaURL", url);
                // Assigning a link to the statistics page to the iframe Html
element.
                iframe.dom.src = url;
            }
```

```
        },
        rules: {}
    };
});
```

Once the schema is saved, you'll notice the [New URL] button in the [Web-stats] section, and the <iframe> element, which will display the stats for the specified Website. (Fig. 7).

Fig. 7. Page editing section record