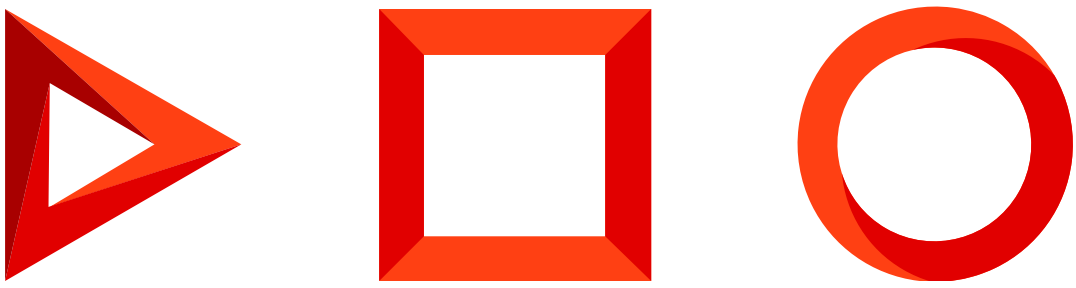
Creatio Academy

Разработка решений в Creatio

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

Разработка приложений на платформе Creatio

Основы

Creatio — это low-code платформа, которая предоставляет широкие возможности для ускорения разработки, внедрения и масштабирования приложений. Платформа построена на открытых принципах кастомизации. Это позволяет создавать приложения разработчикам с разным уровнем квалификации — от бизнес-аналитика до разработчика полного цикла. В зависимости от сложности или типа бизнес-задачи, разработка приложений на платформе Creatio предполагает разные уровни кастомизации.

Уровни кастомизации Creatio

С точки зрения логических уровней взаимодействия архитектура платформы Creatio может быть представлена следующим образом:

При разработке приложений в Creatio необходимо учитывать, что уровень ядра — неизменяемый компонент платформы, а разработка реализуется на уровне конфигурации и базы данных.

База данных

База данных — это уровень физического хранения данных. В базе хранятся не только пользовательские данные, но и настройки приложения, а также настройки прав доступа к приложению.

Платформа Creatio предоставляет инструменты для работы с данными непосредственно из интерфейса приложения. Поэтому не возникает необходимости работы с объектами базы данных напрямую.

Существуют задачи, которые логичнее и быстрее реализуются на уровне базы данных. Пользовательскую бизнес-логику можно реализовать на уровне базы данных с помощью представлений и хранимых процедур. Затем реализованную бизнес-логику можно вызывать в пользовательских конфигурационных элементах.

Ядро

Ядро — неизменяемая часть платформы, которая представляет собой набор библиотек с реализацией базовой функциональности приложения.

Back-end библиотеки реализованы на языке C# с использованием классов платформы .NET Framework. У разработчика есть возможность создавать экземпляры back-end классов и использовать функциональность back-end библиотек, но при этом нельзя вносить изменения в эти классы и библиотеки.

Основные **back-end компоненты ядра**:

- ORM-модель данных и методы работы с ней. В большинстве случаев для доступа к данным рекомендуется использовать именно объектную модель, хотя прямой доступ к базе данных также реализован в back-end компонентах ядра.
- Пакеты и механизм замещения.
- Библиотеки серверных элементов управления. К таким элементам управления относятся страницы, построенные на технологии ASP.NET, которые формируются на сервере, например, страницы раздела [*Конфигурация*] ([*Configuration*]).
- Системные веб-сервисы.
- Функциональность основных дизайнеров и системных разделов.
- Библиотеки для интеграции с внешними сервисами.
- Движок бизнес-процессов (`ProcessEngineService.svc`). Это элемент платформы, который позволяет выполнять алгоритмы, настроенные в виде диаграмм.

Front-end классы ядра реализованы на языке JavaScript с использованием различных фреймворков. Они предназначены для создания пользовательского интерфейса и реализации других бизнес-задач на стороне браузера. Основная **задача** front-end компонентов ядра — обеспечение работы клиентских модулей.

Основные **front-end компоненты ядра**:

- Внешние библиотеки клиентских фреймворков.
- Песочница (sandbox) — специальный компонент клиентского ядра, предназначенный для обеспечения взаимодействия между клиентскими модулями путем обмена сообщениями.
- Базовые модули — файлы на языке JavaScript, в которых реализована функциональность основных объектов платформы.

Конфигурация

Конфигурация — это набор функциональности, который доступен пользователям конкретного рабочего пространства, а именно:

- Серверная логика.
- Автогенерируемые классы, являющиеся продуктом работы настроек приложения.
- Клиентская логика (страницы, кнопки, действия, отчеты, бизнес-процессы и другие настраиваемые конфигурационные элементы).

Конфигурация является легко изменяемой частью приложения. Конкретную конфигурацию формируют

следующие типы элементов:

- **Объекты** — сущности, предназначенные для хранения данных, которые объединяют таблицу в базе данных и класс на серверной стороне.
- **Бизнес-процессы** — настраиваемые элементы, которые представляют собой визуальный алгоритм пользовательских действий.
- **Клиентские модули**.

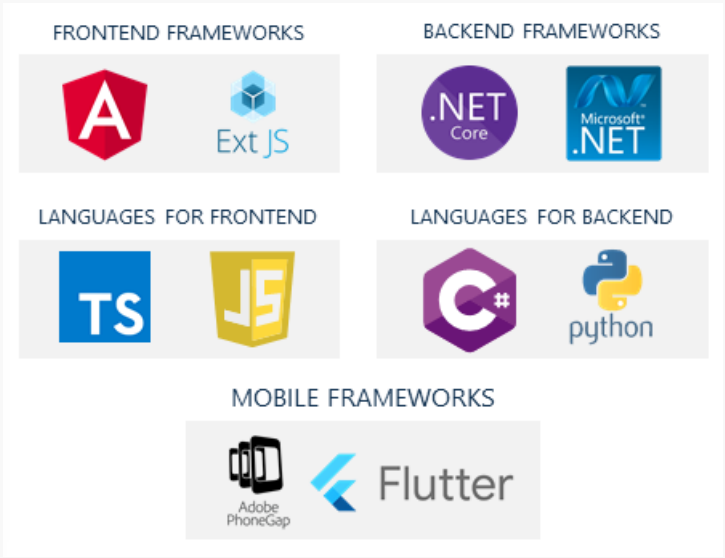
Инструменты разработки приложений

Creatio предоставляет широкий спектр инструментов для создания новых приложений и расширения существующих.

Стек open-source технологий

Поддержка стандартных языков программирования и фреймворков ускоряет разработку, обеспечивает эффективную поддержку и разработку компонентов приложения, а также гарантирует наличие квалифицированных специалистов с необходимым стеком технологий.

Платформа Creatio поддерживает следующие **технологии**:



Для реализации сложной бизнес-логики, интеграции и настройки, Creatio предоставляет **встроенную IDE**, инструменты которой позволяют ускорить решение типовых задач по конфигурированию платформы. Встроенная IDE позволяет использовать C# или JavaScript для реализации следующих **задач**:

- Расширение и изменение функций Creatio.
- Организация взаимодействия с системами контроля версий.
- Передача изменений между средой разработки, предпромышленной и промышленной средами.

Для кастомизации Creatio разработчики могут использовать **стороннюю IDE** (например, Microsoft Visual Studio), которая позволяет работать с проектами в локальной файловой системе. Это ускоряет и

упрощает процесс разработки за счет использования знакомой IDE, наличия множества плагинов, расширений и интеграций с различными инструментами разработки, системами контроля версий и т. д. Также позволяет разработчикам сэкономить время на изучении новых инструментов и сосредоточиться на разработке кода C# и JavaScript в знакомой IDE.

Low-code/no-code разработка

Инструменты low-code/no-code разработки представляют собой набор визуальных drag-and-drop редакторов пользовательского интерфейса. Они позволяют выполнять следующие **задачи**:

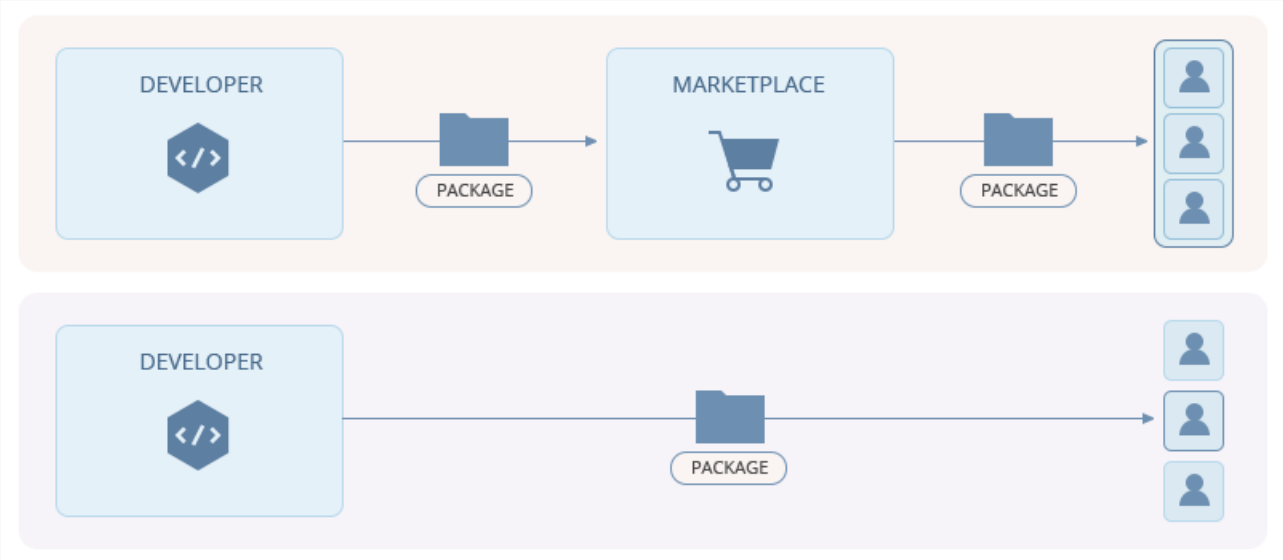
- Автоматизировать бизнес-процессы и динамические кейсы.
- Моделировать структуру данных.
- Моделировать пользовательский интерфейс для веб- и мобильных приложений.
- Создавать информационные панели и отчеты.
- Настраивать интеграции.
- Строить прогнозные модели и т. д.

Большинство вариантов использования покрываются low-code инструментами. Минимальное ручное кодирование, а также легкодоступные встроенные инструменты сделали разработку приложений доступной для бизнес-экспертов, опытных пользователей, аналитиков и citizen developers. Описание no-code инструментов содержится в статье Low-code/no-code.

Механизм пакетов

Независимо от инструмента, используемого для разработки приложения Creatio, все кастомизации упаковываются в пакеты, которые представляют собой ключевой компонент архитектуры Creatio. Пакетная архитектура позволяет выделять целостные функциональные блоки и оформлять их в виде отдельных модулей, управлять иерархией пакетов и версионностью. Это дает возможность быстро расширять конфигурацию, а также переносить изменения между средой разработки, предпромышленной и промышленной средами. Этот же механизм лежит в основе решений, публикуемых на площадке Marketplace.

Любой продукт Creatio — это набор пакетов, которые устанавливаются поверх ядра Creatio. Пакет содержит схемы объектов, исходный код, бизнес-процессы, отчеты и т. д. Он также может содержать сторонние сборки, SQL-сценарии, системные настройки и пользовательские данные.



Модель расширения Creatio основана на **принципе открытости-закрытости**, при котором основная логика приложения закрыта для прямых манипуляций, но открыта для расширения и модификации путем добавления пользовательских пакетов. Каждый пакет может быть расширен пакетом другого издателя (партнера-интегратора, разработчика Marketplace или заказчика). Это позволяет платформе эффективно совмещать out-of-the box продукты, рыночные решения и настройки клиентов практически в любой комбинации.

Архитектура пакетов — это основной механизм доставки и развертывания пользовательских приложений, расширений и шаблонов, полностью интегрированный в Creatio Marketplace. Creatio Marketplace представляет собой экосистему для разработки, распространения и получения кастомизаций — как пользовательских приложений и шаблонов, так и обновлений и изменений отраслевых приложений.

Пакеты не влияют на логику ядра платформы, обеспечивая беспроблемную доставку обновлений для ядра Creatio, а также параллельное развертывание и обновление настраиваемых функций.

Независимая от СУБД архитектура и собственная ORM

Creatio — это **СУБД-независимая платформа**, в основе которой лежит ORM, разработанная Creatio. Это позволяет разработчикам легко создавать и доставлять пользовательские приложения для различных конфигураций в базе данных Oracle, PostgreSQL или MS SQL Server без изменений в кодовой базе.

Бизнес-логика, основанная на процессах

Это среда для взаимодействия между разработчиками полного цикла и бизнес-аналитиками. Используя визуальные drag-and-drop редакторы бизнес-процессов, пользователи могут создавать собственную бизнес-логику, просто нарисовав необходимый процесс. Работа с бизнес-процессами описана в блоке статей Бизнес-процессы.

Интеграции

Платформа Creatio предоставляет необходимые инструменты для интеграции со сторонними системами и приложениями, включая поддержку REST API, протокола OData, SOAP-сервисов, возможности

аутентификации по протоколам OAuth и LDAP. Интеграции можно разрабатывать как часть приложения Creatio или стороннего приложения. Сложные инструменты обеспечивают безопасность данных во время интеграции для идентификации и контроля доступа, а также управления структурой пользователей.

Low-code/no-code

Основы

Платформа Creatio предоставляет широкие возможности low-code/no-code кастомизации: от конфигурирования существующих приложений до разработки пользовательских бизнес-решений.

Low-code/no-code платформы ориентированы на непрофессиональных разработчиков (citizen developers) и используют визуальные интерфейсы с простой логикой и функциями drag-and-drop. **Встроенные инструменты** платформы Creatio дают возможность пользователям, не обладающим знаниями языков программирования или процессов разработки программного обеспечения, создавать свои приложения.

Преимущества low-code/no-code платформ:

- Быстрая разработка приложений.
- Быстрое развертывание.
- Исполнение и управление приложениями.
- Декларативная, высокоуровневая разработка.
- Возможность моделирования данных, разработки интерфейса и бизнес-логики.
- Пользовательское конфигурирование приложений.

Low-code/no-code инструменты

В Creatio реализованы следующие **low-code/no-code инструменты**:

- Дизайнер процессов.
- Мастер разделов.
- Дизайнер кейсов.
- Встроенные инструменты интеграции.
- Технологии машинного обучения.

Дизайнер процессов

Дизайнер процессов — это визуальный редактор для разработки исполняемых бизнес-процессов на основе нотации BPMN 2.0. Исполняемые бизнес-процессы позволяют реализовать пользовательскую бизнес-логику: от автоматизации рутинных задач до разработки сложных интеграций. Дизайнер процессов поддерживает **многопользовательскую разработку**. Использование встроенных элементов бизнес-процессов позволяет выполнять следующие **задачи**:

- Планировать пользовательские действия.
- Работать с интерфейсными страницами.

- Обрабатывать данные.
- Вызывать внешние веб-сервисы и т. д.

Назначение интерфейса настройки и встроенных инструментов валидации:

- Проектирование схемы бизнес-процесса.
- Изменение схемы бизнес-процесса.
- Отладка схемы бизнес-процесса с учетом нюансов выполнения.

Описание работы с бизнес-процессами в Creatio содержится в блоке статей Бизнес-процессы.

Мастер разделов

Мастер разделов позволяет выполнять следующие **задачи**:

- Настраивать пользовательский интерфейс.
- Добавлять и настраивать бизнес-логику.
- Создавать и настраивать разделы, страницы, детали, вкладки и мини-карточки.
- Добавлять поля, изменять их положение или скрывать.

Описание работы с мастером разделов содержится в блоке статей Настройка интерфейса и бизнес-логики.

Дизайнер кейсов

Дизайнер кейсов построен на основе ProcessEngineService и позволяет автоматизировать неструктурированные процессы, ход которых определяется динамически, в соответствии с принятой бизнес-логикой. Бизнес-кейсы состоят из стадий, каждая из которых может включать в себя ряд последовательных или параллельных действий — "шагов", как автоматических, так и пользовательских действий. No-code дизайнер кейсов позволяет выполнять следующие **задачи**:

- Изменять последовательность шагов на стадии процесса.
- Перемещать шаги на другие стадии процесса.
- Изменять последовательность стадий процесса с помощью drag-and-drop.

Встроенные инструменты интеграции

Creatio может интегрироваться с настраиваемыми веб-службами REST только с помощью low-code инструментов. Расширенные возможности интеграции (с использованием инструментов .Net, REST, SOAP, OData, открытого API), а также система администрирования и контроля доступа позволяют быстро и безопасно встраивать Creatio в информационную среду предприятия. На основе настраиваемой бизнес-логики Creatio выполняет следующую **последовательность действий**:

1. Генерирует запрос.
2. Отправляет запрос веб-сервису.
3. Получает ответ.
4. Извлекает необходимые данные.

Данные, полученные от веб-сервиса, можно использовать для создания или обновления записей в базе данных Creatio, а также для реализации специальной бизнес-логики или автоматизации.

Описание способов интеграции содержится в статье Интеграции.

Инструменты машинного обучения

Технологии машинного обучения позволяют автоматизировать процесс принятия решений путем анализа исторических данных и выявления зависимостей между большими объемами информации. В Creatio реализованы следующие **модели** машинного обучения:

- Прогнозирование справочных полей — позволяет автоматически заполнять поле на основании данных текущей записи и решений, принятых пользователями ранее в аналогичных ситуациях.
- Прогнозирование числовых полей — позволяет рассчитать прогнозное значение числового поля.
- Прогнозирование рейтинга записей — позволяет выполнять скоринг (прогнозирование рейтинга) записей в любом разделе системы.
- Рекомендательное прогнозирование — позволяет создавать подборки записей из объектов системы и предлагать их клиентам или партнерам.

Back-end (C#)

Основы

Разработка решений в Creatio предполагает разные уровни кастомизации в зависимости от сложности или типа бизнес-задачи. При этом следует учитывать, что уровень ядра является неизменяемым компонентом системы, а разработка в Creatio реализуется на уровне конфигурации.

Кастомизация приложений на уровне back-end реализуется в направлениях работы с данными, веб-сервисами, настройкой бизнес-логики объектов и т. д.

Для этих целей Creatio предоставляет набор различных инструментов — от разработки схемы [*Исходный код*] ([*Source code*]) или [*Действие процесса*] ([*User Task*]) во встроенной IDE до создания завершенных пользовательских проектов, подключаемых к приложению в качестве внешних библиотек.

Направления back-end разработки

На уровне back-end реализуются следующие направления разработки:

- ORM-модель данных и методы работы с ней.
- Реализация прямого доступа к базе данных.
- Создание и использование системных веб-сервисов.
- Настройка интеграции с внешними сервисами.
- Работа с компонентами системы и микросервисами.
- Расширенная настройка бизнес-процессов и встроенных процессов объектов приложения.
- Разработка бизнес-логики объектов.

ORM-модель данных и прямой доступ к базе данных

ORM-модель данных реализована в системе классами пространства имен `Terrasoft.Core.Entities` :

- `Terrasoft.Core.Entities.EntitySchemaQuery` — построение запросов выборки записей из таблиц базы данных с учетом прав доступа текущего пользователя.
- `Terrasoft.Core.Entities.Entity` — доступ к объекту, который представляет собой запись в таблице базы данных.

В большинстве случаев для доступа к данным рекомендуется использовать именно объектную модель, хотя прямой доступ к базе данных также реализован в back-end компонентах ядра.

Прямой доступ к базе данных предоставляет группа классов back-end ядра приложения из пространства имен `Terrasoft.Core.DB` :

- `Terrasoft.Core.DB.Select` — построение запросов выборки записей из таблиц базы данных.
- `Terrasoft.Core.DB.Insert` — построение запросов на добавление записей в таблицы базы данных.
- `Terrasoft.Core.DB.InsertSelect` — построение запросов на добавление записей в таблицы базы данных.
- `Terrasoft.Core.DB.Update` — построение запросов на изменение записей в таблице базы данных.
- `Terrasoft.Core.DB.Delete` — построение запросов на удаление записей в таблице базы данных.
- `Terrasoft.Core.DB.DBExecutor` — возможность создания и выполнения сложных запросов (например, с несколькими вложенными фильтрациями, различными комбинациями join-ов и т. д.) к базе данных.

Веб-сервисы

Сервисная модель Creatio предоставляет базовый набор веб-сервисов, с помощью которых может быть организована интеграция Creatio с внешними приложениями и системами.

Примеры **системных сервисов**:

- odata — обмен данными с Creatio по протоколу OData 4.
- ProcessEngineService.svc — запуск бизнес-процессов Creatio из внешних приложений.

Кроме системных, в Creatio реализованы **конфигурационные веб-сервисы**, предназначенные для вызова из клиентской части приложения.

Дополнительно к базовым сервисам разработчик имеет возможность создать **пользовательский веб-сервис**, предназначенный для решения узких бизнес-задач конкретного проекта.

Внешние сервисы

В ядре приложения реализованы классы, обеспечивающие интеграцию с внешними сервисами. Например, пространство имен `Terrasoft.Social` предоставляет возможность интеграции с различными социальными сетями.

Компоненты системы и микросервисы

В ядре приложения реализованы классы, предоставляющие возможности для работы с компонентами системы и микросервисами. Например, пространство имен `Terrasoft.Sync` ядра приложения

предоставляет классы встроенного механизма синхронизации с внешними хранилищами данных (Sync Engine), который позволяет создавать, изменять и удалять `Entity` в системе на основании данных из внешних систем и экспортировать данные во внешние системы. Процесс синхронизации осуществляется с помощью класса `SyncAgent`.

Бизнес-процессы и встроенные процессы объектов

Back-end разработка может потребоваться для **настройки сложных бизнес-процессов** (Элемент процесса [*Задание-сценарий*] ([*Script-task*])) либо для создания пользовательских **повторяющихся операций бизнес-процесса** (Конфигурационная схема [*Действие процесса*] ([*User Task*])).

Встроенные процессы объекта могут быть настроены как при помощи no-code инструментов, так и с использованием back-end разработки. Использование программного кода позволяет более гибко настроить поведение объекта в случае наступления определенных событий.

Бизнес-логика объектов

В Creatio есть возможность разрабатывать бизнес-логику объектов без использования событийных подпроцессов. Для этого достаточно создать класс-наследник базового ядрового класса `Terrasoft.Core.Entities.Events.BaseEntityEventListener` и декорировать его атрибутом `EntityEventListener` с указанием имени сущности, для которой необходимо выполнить подписку событий. После чего можно переопределять методы-обработчики нужных событий.

Инструменты и утилитные возможности back-end разработки

Схема исходного кода

Основной возможностью back-end разработки является создание в пользовательском пакете схемы типа [*Исходный код*] ([*Source code*]).

Все изменения схемы, выполняемые в дизайнере, осуществляются в оперативной памяти. Чтобы изменения были сохранены на уровне метаданных схемы, схему следует сохранить. Для того чтобы изменения произошли на уровне базы данных, схему необходимо опубликовать.

Разработку схемы типа [*Исходный код*] ([*Source code*]) можно вести в файловой системе с использованием удобной IDE.

Конфигурирование бизнес-процессов

Включение специфической back-end логики в бизнес-процесс возможно с помощью программного кода, выполняемого элементом [*Задание-сценарий*] ([*Script-task*]).

Во время работы с бизнес-процессами в Creatio часто возникает необходимость выполнять однотипные операции. Для этих целей лучше всего подходит элемент [*Выполнить действие процесса*] ([*User task*]), для которого существует возможность выбрать наиболее подходящий в конкретной ситуации тип действия — [*Пользовательское действие*] ([*User task*]).

По умолчанию в системе доступно множество пользовательских действий, однако могут возникнуть ситуации, когда для выполнения определенного бизнес-процесса необходимо создать новое

пользовательское действие.

Создать новое пользовательское действие можно с помощью конфигурационной схемы [*Действие процесса*] ([*User Task*]). В простой реализации действие процесса частично повторяет логику элемента процесса [*Задание-сценарий*] ([*Script-task*]), однако созданное однажды действие можно использовать многократно в разных процессах. А при внесении изменений в действие процесса, такие изменения сразу же будут применены ко всем процессам, в которых это действие использовалось.

Внешние библиотеки

В структуру пользовательского пакета могут быть включены внешние библиотеки, созданные пользователем. Это позволяет реализовать сложную логику, механизмы наследования, инкапсуляции при разработке специфического проектного решения.

Пакет-проект

Одним из инструментов Creatio для ускорения back-end разработки приложения является пакет-проект. Это пакет, который позволяет разрабатывать функциональность как обычный C#-проект. Новая функциональность в виде скомпилированной библиотеки и *.cs-файлов включается в файловый контент пакета. При старте или перезапуске приложения Creatio собирает информацию о том, что в пакетах есть подготовленные библиотеки и сразу же подключает их в приложение.

Front-end (JS)

Основы

Front-end платформы — это набор модулей, которые определяются и загружаются асинхронно по мере необходимости.

Основой front-end Creatio является ядро.

Уровень ядра предоставляет:

- возможность использования объектно-ориентированного подхода и механизма наследования;
- инструменты для определения и асинхронной загрузки модулей и их зависимостей;
- функциональность базовых элементов системы;
- механизм взаимодействия модулей.

Front-end разработка в Creatio осуществляется на уровне конфигурации и представляет собой создание новых и расширение базовых визуальных и невизуальных модулей, схем моделей представления.

Компоненты front-end ядра приложения

Внешние JS-библиотеки

ExtJS — JavaScript-фреймворк для разработки веб-приложений и пользовательских интерфейсов. В Creatio ExtJS используется как механизм создания структуры классов клиентской части ядра. Позволяет реализовать объектно-ориентированный подход, который в чистом виде не реализован в JavaScript. Предоставляет возможность создавать классы, реализовывать иерархию наследования, группировать классы в пространства имен.

RequireJS — библиотека, которая реализует подход Asynchronous Module Definition (AMD). Подход AMD декларирует механизм определения и асинхронной загрузки модулей и их зависимостей.

Angular — JavaScript-фреймворк для разработки одностраничных приложений. Его цель — расширение браузерных приложений на основе MVC-шаблона, а также упрощение тестирования и разработки. В Creatio реализована возможность встраивания кастомных Angular-компонентов с использованием единого ядра Angular.

Базовые JS-классы

Неизменяемая часть системы. Эти классы:

- Обеспечивают работу клиентских модулей конфигурации.
- Определяют функциональность основных объектов системы, элементов управления, перечислений и констант.
- Хранятся в виде исполняемых файлов на диске в составе папок приложения.

Механизм сообщений

Sandbox — компонент ядра, который служит диспетчером при взаимодействии модулей системы. Sandbox предоставляет механизм обмена сообщениями между модулями (методы `sandbox.publish()` и `sandbox.subscribe()`) и загрузки модулей по требованию в интерфейс приложения (метод `sandbox.load()`).

Асинхронное определение модулей

Front-end платформы имеет **модульную структуру**.

Модуль — инкапсулированный в обособленный блок набор функциональности, который в свою очередь может использовать другие модули в качестве зависимостей.

Создание модулей в JavaScript декларируется паттерном программирования "Модуль". Классическим приемом реализации этого паттерна является использование анонимных функций, возвращающих определенное значение (объект, функцию и т. д.), которое ассоциируется с модулем. При этом значение модуля экспортируется в глобальный объект.

Для управления большим количеством модулей в Creatio загрузка модулей и их зависимостей выполняется в соответствии с подходом **Asynchronous Module Definition** (AMD).

Подход AMD декларирует механизм определения и асинхронной загрузки модулей и их зависимостей, который в процессе работы с системой позволяет подгружать только те данные, которые необходимы для работы в текущий момент. В Creatio для работы с модулями используется загрузчик **RequireJS**.

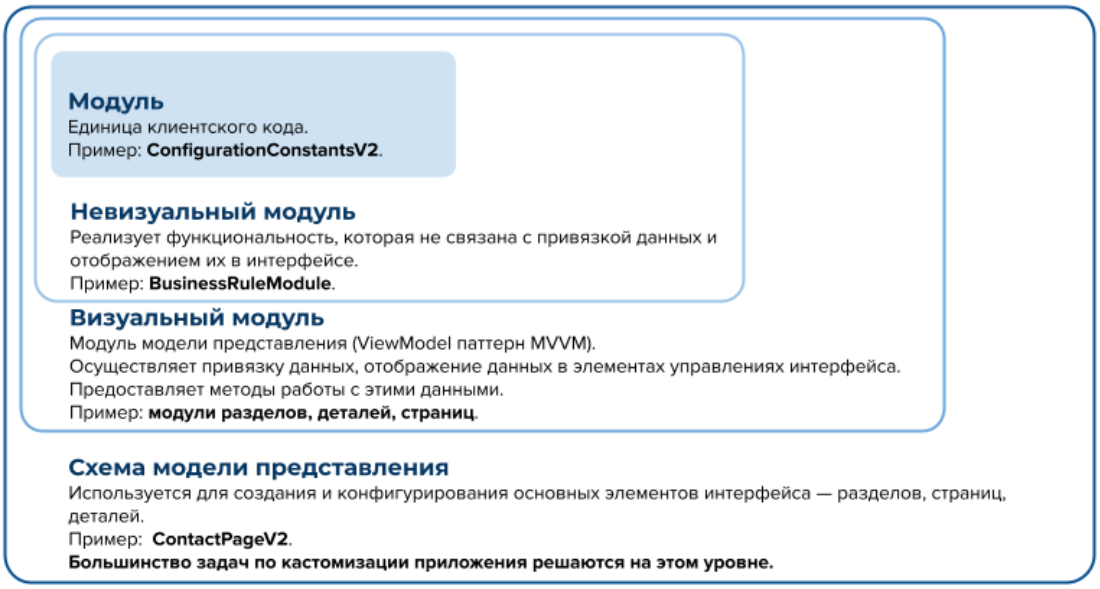
Принципы определения модулей в Creatio:

- Объявление модуля выполняется в специальной функции `define()`, которая регистрирует функцию-фабрику для инстанцирования модуля, но при этом не загружает его немедленно в момент вызова.
- Зависимости модуля передаются как массив строковых значений, а не через свойства глобального объекта.
- Загрузчик выполняет загрузку всех модулей-зависимостей, переданных в качестве аргументов в `define()`. Модули загружаются асинхронно, при этом фактически порядок их загрузки определяется загрузчиком произвольно.
- После того как загрузчиком будут загружены все указанные зависимости модуля, будет вызвана функция-фабрика, которая вернет значение модуля. При этом в функцию-фабрику в качестве аргументов будут переданы загруженные модули-зависимости.

Модульная разработка в Creatio

Реализация всей пользовательской функциональности выполняется в **клиентских модулях**.

Иерархия модулей в Creatio



Несмотря на некоторые функциональные различия, все клиентские модули Creatio имеют одинаковую структуру описания, которая соответствует формату описания модулей AMD.

Общая структура описания клиентского модуля

```
define(
    "ModuleName",
    "dependencies",
    function(dependencies) {
        // someMethods…
        return { moduleObject };
});
```

- `ModuleName` — имя модуля;
- `dependencies` — модули-зависимости, функциональность которых можно использовать в текущем модуле;
- `moduleObject` — конфигурационный объект созданного модуля.

Виды клиентских модулей в Creatio:

- Невизуальный модуль.
- Визуальный модуль.
- Схема модели представления.

Невизуальный модуль

Содержит реализацию функциональности системы, которая не сопряжена с привязкой данных и отображением их в интерфейсе.

Структура описания невизуального модуля

```
define(
    "ModuleName",
    "dependencies",
    function(dependencies) {
        // Методы, реализующие необходимую бизнес-логику.
        return;
});
```

Примерами невизуальных модулей являются утилитные модули, которые реализуют служебные функции.

Визуальный модуль

Используется для создания пользовательских визуальных элементов системы.

Реализует модель представления (ViewModel) согласно шаблону MVVM. Инкапсулирует в себе данные, которые отображаются в элементах управлениях графического интерфейса, а также методы работы с этими данными.

Визуальный модуль содержит **методы**:

- `init()` — метод инициализации модуля.
  Отвечает за инициализацию свойств объекта класса, а также за подписку на сообщения.
- `render(renderTo)` — метод отрисовки представления модуля в DOM.
  Должен возвращать представление.

  Принимает единственный аргумент `renderTo` — элемент, в который будет вставлено представление объекта модуля.
- `destroy()` — метод, отвечающий за удаление представления модуля, удаление модели представления, отписку от ранее подписанных сообщений и уничтожение объекта класса модуля.

Для создания визуального модуля можно использовать базовые классы ядра. Например, создать в модуле класс-наследник `Terrasoft.configuration.BaseModule` или `Terrasoft.configuration.BaseSchemaModule`. Это базовые классы, в которых в необходимой степени уже реализованы необходимые методы визуального модуля — `init()`, `render(renderTo)` и `destroy()`.

Структура описания визуального модуля (наследника базового класса Terrasoft.BaseMo...

```
define("ModuleName", "dependencies", function(dependencies) {
    // Определение класса модуля.
    Ext.define("Terrasoft..configuration.className") {
```

```
        alternateClassName: "Terrasoft.className"
        extend: "Terrasoft.BaseModule",
        …
        // Свойства и методы.
        …
    };
    // Создание объекта модуля.
    return Ext.create(Terrasoft.className)
});
```

Примерами визуальных модулей являются модули, реализующие функциональность элементов управления на странице приложения.

Схема модели представления

Наиболее частыми задачами кастомизации приложения являются создание и доработка основных элементов интерфейса — разделов, страниц, деталей.

Модули этих элементов имеют **шаблонизированную структуру** и называются схемами моделей представления.

Схема модели представления является конфигурационным объектом для генерации представления и модели представления генераторами Creatio `ViewGenerator` и `ViewModelGenerator`.

Для большинства задач кастомизации в Creatio используется механизм замещения базовых схем. К таким схемам относятся все схемы из предустановленных пакетов конфигурации.

Основные базовые схемы моделей представления:

- `BasePageV2`
- `BaseSectionV2`
- `BaseDetailV2`

Структура схемы модели представления

```
define("SchemaName", "dependencies",
    function(dependencies) {
        return {
            entitySchemaName: "ExampleEntity",
            mixins: {},
            attributes: {},
            messages: {},
            methods: {},
            rules: {},
            businessRules: {},
            modules: {},
            diff:
        };
});
```

Примерами схем моделей представления являются схемы страниц, разделов и деталей.

Интеграции

Основы

Интеграция — это обмен данными между системами с возможной последующей обработкой. **Цель** интеграции — автоматический перенос данных, которые внес пользователь, из одной системы в другую.

Платформа Creatio предлагает широкие возможности для интеграции внешних программных продуктов. Открытый API Creatio позволяет создавать интеграционные решения любой сложности.

Creatio поддерживает следующие **способы интеграции**:

- Интеграция внешних приложений с Creatio.
- Интеграция Creatio с внешними приложениями.

Выбор способа интеграции зависит от следующих факторов:

- Потребностей клиента.
- Типа и архитектуры внешних приложений.
- Компетенции разработчика.

Интеграция внешних приложений с Creatio

Задачи интеграции внешних приложений с Creatio:

- Выполнение CRUD-операций с объектами Creatio.
- Запуск бизнес-процессов.
- Реализация пользовательских задач, которые можно решить в рамках открытого API Creatio.

Сервисы работы с данными

Сервисы работы с данными используются для выполнения CRUD-операций с объектами Creatio. В Creatio реализованы следующие сервисы работы с данными:

- Протокол OData.
- Сервис DataService.

Протокол OData

OData (Open Data Protocol) — это утвержденный ISO/IEC стандарт OASIS, который определяет набор лучших практик для построения и использования REST API. Протокол позволяет создавать службы на основе REST, которые с помощью HTTP-запросов предоставляют веб-клиентам возможность публиковать и редактировать ресурсы, идентифицированные с использованием URL и определенные в модели данных.

Приложение Creatio поддерживает протоколы OData 4 и OData 3. OData 4 предоставляет больше возможностей, чем OData 3. Основное **отличие** протоколов — ответ на запрос, возвращаемый сервером, имеет разный формат данных. Различия протоколов OData 3 и OData 4 описаны в официальной документации OData. При планировании интеграции с Creatio по протоколу OData необходимо использовать протокол версии 4.

Детальное описание протокола содержится в документации OData.

Сервис DataService

DataService (разработан Creatio) — сервис, который реализует связь между клиентской и серверной частями платформы. С помощью DataService выполняется передача данных, введенных в пользовательском интерфейсе, в серверную часть приложения для последующей обработки и сохранения в базу данных.

Сервис запуска бизнес-процессов

Системный веб-сервис ProcessEngineService.svc используется для запуска бизнес-процессов из внешнего приложения.

Пользовательский веб-сервис

В конфигурации Creatio существует возможность создавать пользовательские веб-сервисы, которые используются для реализации специфических интеграционных задач. Конфигурационный веб-сервис представляет собой RESTful-сервис, реализованный на базе технологии WCF.

Интеграция Creatio с внешними приложениями

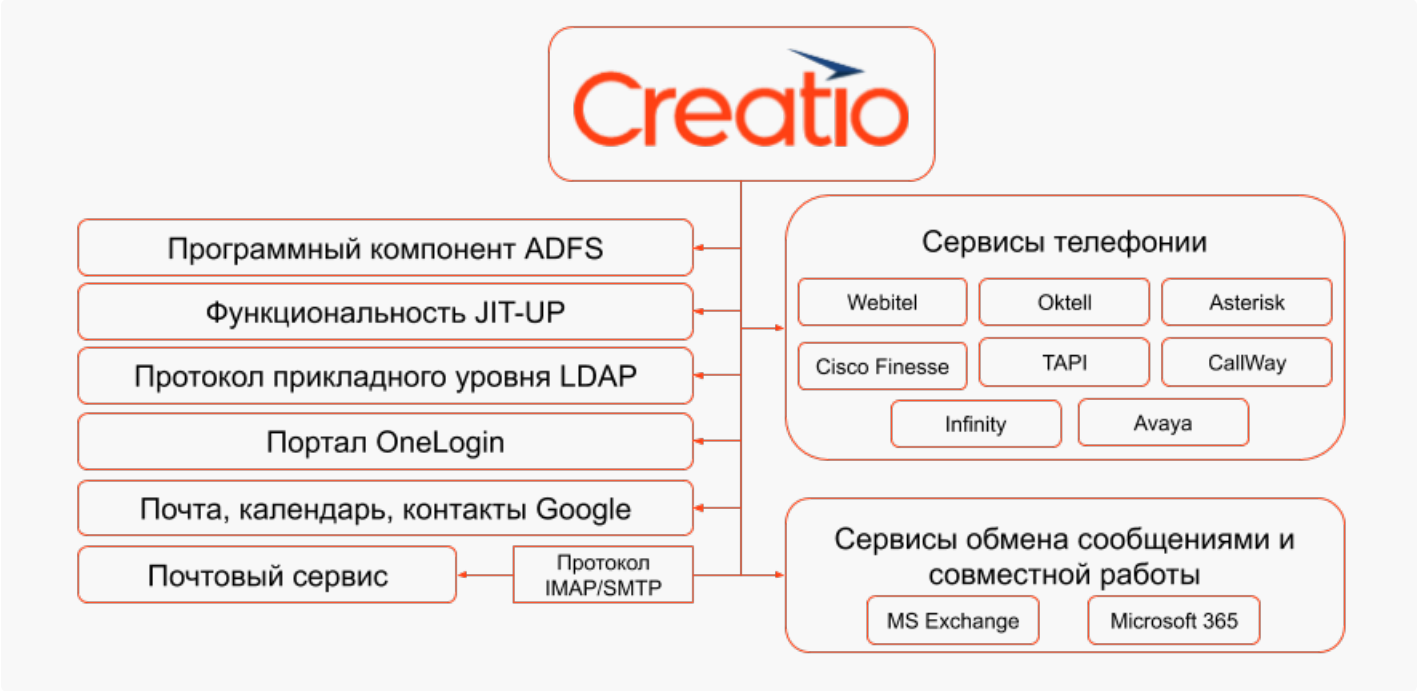
Используя low-code/no-code инструменты интеграции, можно объединить корпоративные приложения в единую цифровую экосистему. Интеграция Creatio с внешними приложениями предполагает разработку или использование готовых интеграционных решений

Разработка интеграционных решений

Используя no-code инструменты, Creatio позволяет настроить интеграцию с пользовательским RESTful API. После настройки интеграции с веб-сервисом его можно вызвать в бизнес-процессе. Инструменты REST API позволяют взаимодействовать со сторонними веб-сервисами без привлечения разработчиков.

Готовые интеграционные решения

Готовые интеграционные решения, которые предоставляет Creatio, представлены ниже.



Creatio предоставляет **готовые интеграционные решения** со следующими внешними приложениями:

- Порталом OneLogin, который используется в качестве единой точки входа для всех сервисов компании.
- Программным компонентом Active Directory Federation Services (ADFS), который используется для управления возможностью единого входа для всех пользователей системы.
- Функциональностью Just-In-Time User Provisioning (JIT UP), которая избавляет от необходимости создания учетных записей для каждого отдельного сервиса и поддержания актуальности базы пользователей вручную.
- Протоколом прикладного уровня Lightweight Directory Access Protocol (LDAP), который обеспечивает доступ к специализированной базе данных, где обычно хранятся учетные данные пользователей, компьютеров и т. д.
- Почтовым сервисом по протоколу IMAP/SMTP.
- Почтой, календарем и контактами Google.
- Сервисами телефонии Webitel, Oktell, Asterisk, Cisco Finesse, TAPI, CallWay, Infinity, Avaya.
- Сервисами обмена сообщениями и совместной работы MS Exchange и Microsoft 365.