Сервисная платформа JPrime

Функциональные характеристики

# Введение

Настоящий документ является общим описанием универсальной сервисной платформы JPrime (далее по тексту — Платформа, JPrime), представляющей собой предметно-ориентированную среду разработки, предназначенную для создания прикладных решений различного масштаба и предметной специфики.

Настоящий документ описывает Платформу в целом.

# Перечень терминов, сокращений и обозначений

Используемые в настоящем документе термины и основные понятия области автоматизированных систем определены в ГОСТ 34.003-90.

Дополнительно в текст введены специальные сокращения, приведенные в Таблица 1:

Таблица 1. Специальные термины и определения

|  |  |
| --- | --- |
| Термин | Определение |
| ABAC | Attribute-based access control (ABAC)  (перевод. управление доступом на основе атрибутов) — это система авторизации, которая определяет доступ на основе атрибутов, связанных с субъектами безопасности, ресурсами и средой. |
| API | Application Programming Interface — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными |
| CRUD-операции | CRUD – это акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete). Введён Джеймсом Мартином в 1983 году как стандартная классификация функций по манипуляции данными. |
| Groovy | объектно-ориентированный язык программирования, разработанный для платформы Java как дополнение к языку Java.  Использует java-подобный синтаксис с динамической компиляцией в JVM байт-код и напрямую работает с другим java-кодом и библиотеками. |
| GUI | GUI (англ. graphical user interface – графический пользовательский интерфейс) — система средств для взаимодействия пользователя с электронными устройствами, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.). |
| HATEOAS | HATEOAS (Hypermedia as the Engine of Application State) — архитектурные ограничения для REST-приложений.  С помощью HATEOAS клиент взаимодействует с сетевым приложением, сервер которого обеспечивает динамический доступ через гипермедиа. REST-клиенту не требуется заранее знать, как взаимодействовать с приложением или сервером за пределом гипермедиа. |
| Java | строго типизированный объектно-ориентированный язык программирования общего назначения. |
| Java-аннотация | специальная форма синтаксических метаданных в языке Java, которая может быть добавлена в исходный код. |
| JDBC | JDBC (англ. Java DataBase Connectivity — соединение с базами данных на Java) — платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE. |
| JSON | JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. |
| JVM | Java Virtual Machine (сокращенно Java VM, JVM) — виртуальная машина Java — основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). |
| JWT | JSON Web Token (JWT) — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности. |
| HTTP | HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных |
| Dead Letter Queue | Dead Letter Queue (DLQ) — это очередь, куда могут перенаправляться сообщения, которые не смогли обработать получатели в обычных очередях. |
| JAR-файл | JAR-файл – это Java-архив (сокращение от англ. Java ARchive). Представляет собой ZIP-архив, в котором содержится часть программы на языке Java. |
| OAuth | открытый протокол (схема) авторизации, обеспечивающий предоставление третьей стороне ограниченный доступ к защищённым ресурсам пользователя без передачи ей (третьей стороне) логина и пароля. OAuth 2.0 - последняя версия, опубликованная в октябре 2012 года. |
| RBAC | Role-based access control (RBAC)  (перевод. управление доступом на основе ролей) – это система авторизации, которая определяет доступ к ресурсам на основе ролей. |
| REST | REST (Representational state transfer) — это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами |
| SOAP | SOAP (от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. |
| SQL | SQL (аббр. от англ. Structured Query Language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. |
| XML | XML (аббр. от англ. eXtensible Markup Language) — «расширяемый язык разметки». Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML - язык с простым формальным синтаксисом, удобный для создания и обработки документов как программами, так и человеком. |
| ГАР | Государственный адресный реестр |
| БД | База данных |
| Платформа, JPrime | Программное обеспечение «Универсальная сервисная платформа JPrime» |
| ПО | Программное обеспечение |
| СУБД | Система управления базами данных |

# Описание платформы JPrime и её назначение

Платформа JPrime представляет собой фреймворк, спроектированный на современном стеке технологий, микросервисной архитектуре с использованием открытого исходного кода и включающий интегрированный набор средств, инструментов и технологий, необходимых для быстрой разработки, внедрения и поддержки прикладных решений автоматизации бизнес-процессов предметной области.

Платформа является кроссплатформенной и поддерживает работу с основными типами операционных систем (Microsoft Windows, Linux, Sun Solaris) и СУБД (PostgreSQL, MySQL, Oracle, MS SQL Server).

Платформа имеет сервис-ориентированную модульную архитектуру, базирующуюся на независимых программных модулях, реализующих логически выделенные прикладные задачи. Каждый модуль строится на единой системной концепции, но при этом изолирован от других модулей и может использовать иной стек технологий и язык программирования.

Платформа построена по классической трехуровневой модели: клиент, сервер приложений и хранилище данных. В качестве реляционных хранилищ данных может использоваться любая реляционная СУБД. Взаимодействие между модулями платформы осуществляется по документированному API. Основным принципом взаимодействия между модулями является лишенный недостатка ожидания немедленного ответа асинхронный обмен, что снижает нагрузку и увеличивает количество обрабатываемых соединений.

Взаимодействие платформы с клиентами осуществляется с помощью REST, JSON, SOAP и прочих stateless-запросов (на стороне сервера не хранятся данные о результатах предыдущих запросов).

Серверная реализация полностью не зависит от клиентской, что позволяет взаимодействовать с любыми типами клиентов.

Таким образом архитектура JPrime является модульной, что позволяет создавать из имеющихся независимых модулей, как монолитное приложение, так и микросервисное, поддерживающее дублирование высоконагруженных узлов.

Однако приоритетным подходом при реализации прикладного решения является сервис-ориентированная архитектура, в которой временная неработоспособность отдельного модуля не влияет на функционирование других частей системы.

Благодаря модульной компоновке и единому подходу при реализации отдельных модуль, JPrime позволяет создавать бизнес-приложения различного направления и функциональности.

Функциональные модули платформы написаны на языке программирования Java.

# Используемый стек технологий и стороннего программного обеспечения

В работе Платформы используется программное обеспечение (далее – ПО) и задействован стек технологий, перечисленные ниже:

|  |  |  |
| --- | --- | --- |
| Название модули | ПО | Примечание |
| Базовый фреймворк на Java | Spring | Постоянно развивающийся Java фреймворк с открытым исходным кодом |
| Реляционная СУБД | PostgreSQL | СУБД с открытым исходным кодом |
| Шина данных | Apache Kafka | Отказоустойчивая, масштабируемая платформа обмена сообщениями |
| Облачное файловое хранилище S3 | Minio | Распределенное, отказоустойчивое, легко расширяемое хранилище, работающее по протоколу S3 |
| Сбор и отображение статистических данных, в том числе логов | ELK (elasticsearch+logstash+kibana) | Программный комплекс, позволяющий собирать информацию из различных источников и агрегировать. Предоставляет визуальный интерфейс для отображения и поиска |
| Регистрация сервисов | Eureka | Отказоустойчивый менеджер сервисов |
| Балансировка и перенаправление запросов | Spring Cloud GateWay + Ribbon | Программный комплекс, позволяющий обеспечить единую точку входа в высоконагруженном приложении |
| Компоновка и развертывание модуль | Docker | Упрощение настройки и взаимодействия различных модулей |
| Среда выполнения Docker | Kubernetes | Система виртуализации, позволяющая объединять ресурсы физических серверов и среда запуска докер-контейнеров |

# Перечень функций платформы JPrime

В качестве основных функций и технических возможностей платформы JPrime можно выделить:

* Обеспечение функций авторизации, аутентификации и криптографии;
* Создание, настройка и управление структуры метаданных приложения через административную консоль;
* Хранение данных с использованием БД различных производителей;
* Обработка данных (задачи, утилиты, работа с экранными формами);
* Управление представлением информации (автоматическая генерация форм работы с объектами, средства разработки GUI);
* Журналирование событий и мониторинг;
* Обеспечение преобразования объектно-ориентированного представления объектов в реляционные структуры и обратно;
* Механизмы создания и управления отчетностью (rtf, txt, xls, dbf);
* Работа с файловой системой;
* Использование цветовых схем;
* Использование языковых библиотек.

При построении архитектуры JPrime принимался во внимание широкий круг задач:

* легкое расширение технических мощностей, используемых платформой;
* простота настройки взаимодействия модулей;
* возможность масштабирования и подключения новых модулей без нарушения работоспособности системы;
* обеспечение работоспособности под нагрузкой и с большим количеством пользователей;
* постоянный мониторинг всего приложения;
* создание единой точки сбора и обработки логов и другой статистической информации.

# Структура платформы Jprime

Платформа JPrime состоит из следующих независимых пакетов модулей:

* Пакет базовых модулей «JPrime-core»;
* Пакет модулей метаописания данных «JPrime-meta»;
* Пакет модулей авторизации «JPrime-auth»;
* Пакет модулей поддержки криптографических функций «JPrime-crypto»;
* Пакет модулей работы с хранилищами «JPrime-storages»;
* Пакет модулей безопасности «JPrime-security»;
* Пакет модулей работы с файлами «JPrime-files»;
* Пакет модулей работы с файловой системой «JPrime-fs»;
* Пакет модулей балансировки запросов «JPrime-routes»;
* Пакет модулей для работы с историей запросов «JPrime-requesthistory»;
* Пакет модулей работы с отчётностью «JPrime-reports»;
* Пакет модулей работы с утилитами «JPrime-utils»;
* Пакет модулей работы с адресами «JPrime-address».

Перечень подключаемых модулей JPrime может меняться в зависимости от требований к функциям конкретного прикладного решения.

Более подробное описание каждого пакета и модулей, из которых он состоит приведено в разделах ниже.

# Пакет базовых модулей «JPrime-core»

Платформа JPrime состоит из ряда основных модулей, условно объединенных в базовый пакет «JPrime-core». Данные модули являются «фундаментом» платформы, основой для наращивания функций и для разработки дополнительных модулей как самой Платформы, так и основанных на ней прикладных решений.

Полный перечень модулей базового пакета Платформы «JPrime-core» приведен в Таблица 2.

Таблица 2. Состав пакета базовых модулей «JPrime-core»

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-common-core | Содержит глобальные константы и API для межсервисного взаимодействия |
| jprime-common-api | Содержит базовое API |
| jprime-common-exceptions | Содержит базовые исключения |
| jprime-security-core | Содержит константы базовых ролей и API для работы с авторизацией пользователя |
| jprime-eureka-client | Модуль подключения клиентов к Eureka Server |
| jprime-fonts | Модуль, содержащий шрифты |
| jprime-groovy-core | Модуль выполнения groovy-скриптов |
| jprime-actuator | Модуль мониторинга состояния сервиса |
| jprime-monitoring-core | Модуль мониторинга состояния приложения |

## Модуль «jprime-common-core»

Модуль «jprime-common-core» содержит глобальные константы и API для межсервисного взаимодействия. Также модуль содержит логику работы с системными и глобальными событиями, а также механизм создания динамических слушателей брокера сообщений Apache Kafka и логику базовой обработки ошибок.

Реализация передачи событий в виде поддержки обмена событиями через Apache Kafka рекомендуется для микросервисных сборок, а поддержка обмена событиями внутри приложения – для монолитных сборок.

### Работа с системными событиями

Системными называются события, регистрируемые в рамках прикладных подсистем. Все системные события в системе являются асинхронными. События инициируются и обрабатываются программным кодом и используются для передачи информации между сервисами.

Модуль «jprime-common-core» предоставляет базовые возможности для разработки дополнительных классов-обработчиков с такими стандартными методами обработки события, как:

* инициализация системного события;
* подписывание на системное событие;
* передача системных событий.

### Работа с глобальными событиями

Глобальные события — это события для общения между разными подсистемами. Глобальное событие может быть инициировано в любом сервисе любой подсистемы.

Модуль «jprime-common-core» реализует возможность производить:

* инициализацию глобального событий;
* подписывание на глобальное событие;
* адресацию глобальных событий.

### Динамические слушатели

В модуле реализован механизм, который позволяет динамически создавать, масштабировать, запускать, останавливать слушателей. В том числе создавать каскады слушателей, задав основной топик и настройки каскада. За это отвечает реализация Dead Letter Queue, которая решает проблему необходимости динамического создания N слушателей согласно настройкам.

### Базовая обработка ошибок

В модуле реализована общая обработка любой ошибки (исключения) и приведение её к универсальному формату с возможностью вывода специфичной детализации в зависимости от кода ошибки.

## Модуль «jprime-common-api»

Модуль «jprime-common-api» содержит базовое API для работы со специфичными JPrime-форматами, такими как:

* диапазон дат,
* диапазон целочисленных,
* простая дробь,
* денежное,
* JSON,
* и др.

(описание специфичных JPrime-форматов см. в разделе «Типы атрибутов» модуля «jprime-common-starter» в Пакете модулей метаописания данных «JPrime-meta»).

Также модуль «jprime-common-api» реализует API для логирования, API отправки синхронных REST-запросов и содержит параметры приложения.

К модулю «jprime-common-api» зависимостями подключены модули базовых исключений «jprime-common-exceptions» и модуль базовой безопасности «jprime-security-core» - см. ниже.

## Модуль «jprime-common-exceptions»

Модуль «jprime-common-exceptions» содержит перечень базовых ошибок (исключений) Платформы. Подключается к модулю «jprime-common-api».

## Модуль «jprime-security-core»

Модуль «jprime-security-core» реализует базовую безопасность системы и содержит константы базовых ролей и API для работы с авторизацией пользователя.

## Модуль «jprime-eureka-client»

Модуль «jprime-eureka-client» предназначен для подключения клиентов к менеджеру сервисов Eureka Server при микросервисной архитектуре.

## Модуль «jprime-fonts»

Модуль «jprime-fonts» отвечает за возможность добавления новых шрифтов через ресурсы jar-файлов.

## Модуль «jprime-groovy-core»

Модуль «jprime-groovy-core» предназначен для выполнения groovy-скриптов, а также предусматривает выполнение groovy-кода в безопасном режиме, без доступа к системным ресурсам.

## Модуль «jprime-actuator»

Модуль мониторинга состояния сервиса «jprime-actuator» основан на Spring Boot Actuator. Кроме стандартной информации, публикуемой Spring Boot Actuator, может также публиковать следующую информацию о сборке:

* дата сборки;
* версия JPrime;
* проектная версия.

## Модуль «jprime-monitoring-core»

В Платформе присутствует функция мониторинга состояния приложения, реализованная с помощью модуля «jprime-monitoring-core».

В процессе мониторинга памяти JVM происходит периодическое сравнение выделенной и используемой памяти приложением, согласно cron-настройке. По умолчанию, каждую минуту при превышении лимита используемой в лог записываются предупреждения.

# Пакет модулей метаописания данных «JPrime-meta»

Пакет модулей метаописания данных «JPrime-meta» содержит логику взаимодействия с данными через метаописание и состоит из модулей, представленных в Таблица 3.

Таблица 3. Модули в составе с пакета модулей метаописания данных

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-common-starter | Содержит типовую логику, API и базовые реализации основных функций системы |
| jprime-reactive-starter | Содержит типовую логику, API и базовые реализации реактивных функций системы |
| jprime-dynamicmeta-core | Базовый модуль для динамической меты |
| jprime-dynamicmeta-core-jdbc | Базовый модуль для динамической меты в БД |
| jprime-dynamicmeta-editor-starter | API и REST-методы редактирования метаданных |
| jprime-dynamicmeta-editor-jdbc | Редактирования метаданных, хранящихся в БД |
| jprime-dynamicmeta-editor-imex | Утилиты экспорт/импорт настроек меты |
| jprime-dynamicmeta-loader-starter | API для загрузчика меты |
| jprime-dynamicmeta-loader-jdbc | Реализация загрузчика меты из БД |
| jprime-meta-rest | Модуль, предоставляющий REST API для публикации метаописания |
| jprime-common-rest | Модуль, предоставляющий REST API для работы с данными на основе метаописания |
| jprime-jpobjecthistory-storage-starter | Содержит API сохранения истории изменения объекта |
| jprime-jpobjecthistory-storage-jdbc | Модуль сохранения истории изменения объекта в БД |
| jprime-jpobjecthistory-storage-rest | Модуль чтения истории изменения объекта |
| jprime-signer-starter | Модуль цифровой подписи данных JPrime |
| jprime-signer-rest | Модуль, предоставляющий REST API для цифровой подписи данных |
| jprime-signer-stamped-starter | Модуль обработки запроса на постановку штампа подписи на файл JPrime |
| jprime-common-codegen | Модуль транформации представления метаописания |

В платформе JPrime выделяются понятия статического и динамического метаописания. Статическое метаописание подгружается программно через Java-аннотации или xml-файлы и не подлежит изменению. Динамическое метаописание подразумевает возможность создания и редактирования метаданных через подключаемый к платформе пользовательский интерфейс. Модули для работы с динамической метой имеют кодовые имена, начинающиеся с «jprime-dynamicmeta-\*» и описаны ниже. Динамическая мета также может быть преобразована в статическую с помощью модуля «jprime-common-codegen».

## Модуль «jprime-common-starter»

Модуль «jprime-common-starter» является одним основных в Платформе и содержит типовую логику работы с метаданными, API для взаимодействия с объектами через метамодель и кодовые имена классов/атрибутов, а также базовые реализации основных функций Платформы, таких как:

* загрузка метаописания;
* заполнение типовых атрибутов;
* загрузка описания маппинга;
* действия с объектами меты;
* управление доступом;
* валидация;
* расчет значений по умолчанию;
* дополнение значений;
* определение дополнительных сведений по объекту;
* поддержка разных форматов JSON для хранения и отображения данных;
* поддержка получения данных через Java-код;
* поддержка получения данных через SQL;
* вызов обработчика CRUD-операций над объектами»;
* работа с событиями.

Подробнее каждая из этих функций рассмотрена ниже.

### Функция «Загрузка метаописания»

Данная функция модуля отвечает за описание структуры данных и их взаимосвязей, позволяет статически или динамически описывать классы, их свойства (атрибуты) и связи между ними. Для задания статического метаописания модуль поддерживает два способа: через java-аннотацию или через xml-файл определенной структуры. Для работы с динамическими изменениями метаописания дополнительно подключаются модули «jprime-dynamicmeta-\*».

#### Способы статического метаописания

##### Java-аннотация

Аннотация описания метакласса должна подчиняться структуре @JPClass в приведенном ниже примере и содержать вложенные аннотации, описывающие структуру атрибутов - @JPAttr.

Пример:

@JPClass(

guid = "cc53f898-ceec-49b3-81a3-43eb3fdc43f0",

code = "userEvent",

qName = "common.userEvent",

name = "Таблица с информацией о пользовательских событиях",

shortName = null,

description = null,

jpPackage = "adminAccess",

attrs = {

@JPAttr(

guid = "a6f40c06-e41f-4003-a4df-a72d304adc7e",

code = "eventGuid",

type = JPType.UUID,

identifier = true,

qName = "common.userevent.eventGuid",

name = "Гуид события",

shortName = null,

description = null

),

@JPAttr(

guid = "a6f40c06-e41f-4003-a4df-a72d304adc7d",

code = "event",

type = JPType.JSON,

qName = "common.userevent.event",

name = "JSON-описание события",

shortName = null,

description = null,

jpProps = {

@JPProperty(

code = "event\_desc",

type = PropertyType.ELEMENT,

qName = "common.userevent.prop",

name = "JSON-описание события",

shortName = null,

description = null,

schemaCode = "test\_schema"

)

},

schemaProps = {

@JPPropertySchema(

code = "test\_schema",

jpProps = {

@JPProperty(

code = "event\_string",

type = PropertyType.STRING,

qName = "common.userevent.string",

name = "поле-описание события",

shortName = null,

description = null

)

}

)

}

)

}

)

##### XML

Для загрузки метаописания через xml файлы должны подчиняться структуре, приведенной в примере ниже:

<?xml version="1.0" encoding="utf-8"?>

<jpClasses>

<jpClass>

<guid>cc53f898-ceec-49b3-81a3-43eb3fdc43f0</guid>

<code>userEvent</code>

<qName>common.userEvent</qName>

<name>Таблица с информацией о пользовательских событиях</name>

<jpAttrs>

<jpAttr>

<guid>a6f40c06-e41f-4003-a4df-a72d304adc7e</guid>

<code>eventGuid</code>

<type>uuid</type>

<length>0</length>

<identifier>true</identifier>

<qName>common.userevent.eventGuid</qName>

<name>Гуид события</name>

<jpProps>

<jpProperty>

<code>testprop</code>

<name>Test Prop</name>

<qName>egu.eguApplication.jsonattr.jsonprop</qName>

<type>element</type>

<jpProps>

<jpProperty>

<code>innertestprop</code>

<name>Inner Test Prop</name>

<qName>egu.eguApplication.jsonattr.injsonprop</qName>

<type>string</type>

</jpProperty>

</jpProps>

</jpProperty>

</jpProps>

</jpAttr>

</jpAttrs>

</jpClass>

</jpClasses>

#### Свойства класса и атрибута

Независимо от способа задания метаописания для класса могут быть заданы свойства, приведенные в Таблица 4, а для атрибута – в Таблица 5.

Таблица 4. Свойства классов

| Свойство | Описание | Обязательность |
| --- | --- | --- |
| guid | Глобальный идентификатор | - |
| code | Кодовое имя класса | + |
| qName | Полный код класса | - |
| name | Название класса | + |
| shortName | Короткое название класса | - |
| description | Описание класса | - |
| jpPackage | Настройки доступа | - |
| attrs | Список атрибутов класса | + |

Таблица 5. Свойства атрибутов

| Свойство | Описание | Обязательность |
| --- | --- | --- |
| guid | Глобальный идентификатор | + |
| code | Кодовое имя атрибута | + |
| qName | Полный код атрибута | - |
| name | Название атрибута | + |
| shortName | Короткое название атрибута | - |
| description | Описание атрибута | - |
| jpPackage | Настройки доступа | - |
| identifier | Признак идентификатора | - |
| mandatory | Признак обязательности | - |
| type | Тип атрибута | + |
| refJpClass | Кодовое имя класса, на который ссылается | - |
| refJpAttr | Кодовое имя атрибута ссылочного класса | - |
| refJpFile | Описание файлового атрибута | - |
| simpleFraction | Описание простой дроби | - |
| money | Описание денежного типа | - |
| virtualReference | Путь виртуальной ссылки | - |
| virtualType | Тип виртуальной ссылки | - |
| length | Длина (для строковых полей, в том числе для строковых виртуальных) | - |
| jpProps | Свойства псевдо-меты | - |
| schemaProps | Схемы вложенных свойств псевдо-меты | - |

#### Типы атрибутов

В качестве типов атрибутов Платформа поддерживает использование следующих вариантов:

| Код типа атрибута | Java тип | Наименование |
| --- | --- | --- |
| uuid | UUID | Глобальный идентификатор |
| string | String | Строка |
| boolean | Boolean | Да/Нет |
| date | LocalDate | Дата |
| time | LocalTime | Время |
| timestamp | LocalDateTime | Полная дата (c учетом часового пояса) |
| datetime | LocalDateTime | Полная дата (без учета часового пояса) |
| integer | Integer | Целочисленное (32 бита) |
| long | Long | Целочисленное (64 бита) |
| biginteger | BigInteger | Длинное целочисленное |
| float | Float | Вещественное (32 бита) |
| double | Double | Вещественное (64 бита) |
| file | String | Файл |
| xml | String | XML-элемент |
| json | String | JSON |
| dateRange | JPDateRange | Диапазон дат |
| tsRange | JPDateTimeRange | Диапазон дат со временем |
| int4range | JPIntegerRange | Диапазон целочисленный (32 бита) |
| money | JPMoney | Денежное |
| simpleFraction | JPSimpleFraction | Простая дробь |
| backReference | - | Обратная ссылка |
| virtualReference | - | Виртуальное значение |

Ниже приведено краткое описание специфичных для JPrime платформы типов атрибутов, таких как file, xml, json, dateRange, tsRange, int4range, money, simpleFraction, virtualReference, backreference, а также объяснена реализация типа «прямая ссылка (ссылка Nx1)».

##### Тип «Файл» (file)

Физически атрибут типа «Файл» (file) является строкой и в нём хранится имя файла в целевом хранилище, а для полноценной работы атрибута необходимо корректное заполнение refJpFile свойства, где:

| Код | Описание | Обязательное |
| --- | --- | --- |
| storageCode | Код ФС хранилища с файлом | + |
| storageFilePath | Путь в ФС хранилище с файлом | + |
| storageCodeAttrCode | Кодовое имя атрибута для хранения кода ФС хранилища с файлом | опционально |
| storageFilePathAttrCode | Кодовое имя атрибута для хранения пути в ФС хранилище с файлом | опционально |
| fileTitleAttrCode | Кодовое имя атрибута для хранения исходного имени файла | опционально |
| fileExtAttrCode | Кодовое имя атрибута для хранения расширения файла | опционально |
| fileSizeAttrCode | Кодовое имя атрибута для хранения размера файла (в байтах) | опционально |
| fileDateAttrCode | Кодовое имя атрибута для хранения даты формирования файла | опционально |
| fileInfoAttrCode | Кодовое имя атрибута для хранения дополнительной информации о файле | опционально |

Пример через xml:

<jpAttr>

<code>scanCopy</code>

<type>file</type>

<qName>jpClass.className1.PersonalCard.scanCopy</qName>

<description>Скан копия гражданина</description>

<refJpFile>

<storageCode>fileStorage</storageCode>

<storageFilePath>filePath</storageFilePath>

<storageCodeAttrCode>attr1</storageCodeAttrCode>

<storageFilePathAttrCode>attr2</storageFilePathAttrCode>

<fileTitleAttrCode>attr4</fileTitleAttrCode>

<fileExtAttrCode>attr5</fileExtAttrCode>

<fileSizeAttrCode>attr6</fileSizeAttrCode>

<fileDateAttrCode>attr7</fileDateAttrCode>

<fileInfoAttrCode>attr8</fileInfoAttrCode>

</refJpFile>

</jpAttr>

##### Тип «XML-элемент» (xml)

Тип «XML-элемент» аналогичен типу «файл», но предназначен для хранения именно файлов с расширением xml для возможности последующей специфичной обработки содержимого атрибута данного типа.

##### Тип «JSON» (json)

Тип «JSON» аналогичен типу «файл», но предназначен для прицельного хранения json-файлов и обеспечения возможности последующей специфичной обработки содержимого атрибута данного типа.

##### Тип «Диапазон дат» (dateRange)

Тип «Диапазон дат» предназначен для хранения пары взаимосвязанных дат, первая из которых не может быть больше второй.

##### Тип «Диапазон дат со временем» (tsRange)

Тип «Диапазон дат со временем» является по смыслу аналогом типа «Диапазон дат», подходит для случаев, когда важно зафиксировать период из дат вместе со временем.

##### Тип «Диапазон целочисленный» (int4range)

Тип «Диапазон целочисленный» предназначен для хранения пары взаимосвязанных целочисленных значений (integer), первое из которых не может быть больше второго.

##### Тип «Денежное» (money)

Физически в атрибуте типа «Денежное» (money) хранится размер, но для полноценной работы атрибута необходимо корректное заполнение money свойства, где:

| Код | Описание | Обязательное |
| --- | --- | --- |
| currencyCode | Код валюты | + |

Пример через xml:

<jpAttr>

<guid>69269c0d-e8a6-49bc-aebe-f71e4242640c</guid>

<code>moneyAttr</code>

<type>money</type>

<qName>jpClass.className1.PersonalCard.moneyAttr</qName>

<description>Деньги</description>

<money>

<currencyCode>RUR</currencyCode>

</money>

</jpAttr>

##### Тип «Простая дробь» (simpleFraction)

Физически в атрибуте типа «Простая дробь» (simpleFraction) хранится числитель, а для полноценной работы атрибута необходимо корректное заполнение simpleFraction свойства, где

| Код | Описание | Обязательное |
| --- | --- | --- |
| integerAttrCode | Кодовое имя атрибута для хранения целого значения | опционально |
| denominatorAttrCode | Кодовое имя атрибута для хранения знаменателя | опционально |

Пример через xml:

<jpAttr>

<guid>69269c0d-e8a6-49bc-aebe-f71e4142640c</guid>

<code>simpleFract</code>

<type>simpleFraction</type>

<qName>jpClass.className1.PersonalCard.simpleFract</qName>

<description>Доля</description>

<simpleFraction>

<integerAttrCode>integerAttr</integerAttrCode>

<denominatorAttrCode>denominatorAttr</denominatorAttrCode>

</simpleFraction>

</jpAttr>

##### Тип «Прямая ссылка (Nx1)»

Прямая ссылка не реализуется отдельным типом, а расширяет свойства атрибута полями refJpClass и refJpAttr. Тип атрибута указывается один из простых (строка, целочисленное и т.д.), но важно чтобы тип поля соответствовал типу атрибута класса, указанного в настройках refJpClass и refJpAttr (кодовое имя класса и кодовое имя атрибута соответственно)

##### Тип «Обратная ссылка» (backreference)

Обратная ссылка не является полноценным свойством объекта: у него нет значения, хранящегося отдельно. Это скорее визуальная настройка, позволяющая вывести все объекты другого класса, связанные с текущим объектом.

Для полноценной работы обратной ссылки необходимо указать refJpClass и refJpAttr (кодовое имя класса и кодовое имя атрибута соответственно), где указанный refJpAttr - является прямой ссылкой на идентификатор текущего класса.

##### Тип «Виртуальное значение» (virtualReference)

| Код | Java тип | Описание |
| --- | --- | --- |
| virtualReference | - | Глобальный идентификатор |

Тип «Виртуальное значение» (virtualReference) позволяет отобразить значение связанного объекта, как свойство текущего объекта. При этом для построения виртуальной ссылки необходима прямая ссылка на класс, свойство которого хотим виртуализировать.

Для полноценной работы виртуальной ссылки необходимо также указать virtualReference (путь виртуальной ссылки) и virtualType (тип итогового значения виртуальной ссылки) в свойствах атрибута.

Пример:

sprWorker

| Код | Описание |
| --- | --- |
| ouid | Идентификатор |
| fio | Фамилия/Имя/Отчество |

sprOrg

| Код | Описание |
| --- | --- |
| ouid | Идентификатор |
| name | Название |
| director | Ссылка на sprWorker.ouid |
| directorFIO | Виртуальное значение на sprWorker.fio |

, где

<jpAttr>

<code>directorFIO</code>

<type>virtualReference</type>

<virtualReference>director.fio</virtualReference>

<virtualType>string</virtualType>

<length>100</length>

<description>ФИО директора</description>

</jpAttr>

### Функция «Заполнение типовых атрибутов»

В модуле «jprime-common-starter» реализована функция автоматического заполнения набора типовых атрибутов благодаря зарезервированным кодовым именам. Их кодовые имена, описание и логика заполнения представлены в Таблица 6.

Таблица 6. Типовые атрибуты

| Кодовое имя атрибута | Описание | Примечание |
| --- | --- | --- |
| userOwnerId | Пользователь, создавший объект | Заполняется userId при создании |
| creationDate | Дата создания | Заполняется датой создания |
| userEditorId | Пользователь, изменивший объект | Заполняется userId при создании и изменении |
| changeDate | Дата редактирования | Заполняется датой создания и изменения |
| jpPackage | Настройка доступа к объекту | Содержит код RBAC настройки доступа к объекту |

### Функция «Загрузка описания маппинга»

Данная функция модуля отвечает за описание места хранения данных, структура которых описана через метаданные, и определяется с помощью описания маппинга.

Структура описания маппинга класса приведена в Таблица 7.

Таблица 7. Описание маппинга класса

| Свойство | Описание | Обязательное |
| --- | --- | --- |
| code | Кодовое имя класса | + |
| storage | Кодовое имя хранилища | + |
| map | Мап на хранилище | + |
| schema | Схема в хранилище | - |
| attrs | Список маппинга атрибутов класса | + |

Структура описания маппинга атрибута приведена в Таблица 8.

Таблица 8.Описание маппинга атрибута

| Свойство | Описание | Обязательное |
| --- | --- | --- |
| code | Кодовое имя атрибута | + |
| map | Мап на хранилище | + |
| fuzzyMap | Мап на поле с индексами нечеткого поиска | - |
| cs | Регистр значений | - |
| readOnly | Запрет на изменение значений | - |

#### Способы описания маппинга

Аналогично вводу метаописания Платформа поддерживает два способа описания маппинга: java-аннотации и xml-файлы определенной структуры.

##### Java-аннотация

Аннотация описания маппинга класса должна подчиняться структуре @JPClassMap в приведенном ниже примере и содержать вложенные аннотации, описывающие маппинг атрибутов - @JPAttrMap.

@JPClassMap(

code = "userEvent",

storage = "userevents",

map = "userevent",

attrs = {

@JPAttrMap(

code = "eventGuid",

map = "eventguid"

)

}

)

##### XML

Для загрузки описания маппинга через xml файлы должны подчиняться структуре, приведенной в примере ниже:

<?xml version="1.0" encoding="utf-8"?>

<jpClassMaps>

<jpClassMap code="userEvent" storage="userevents" map="userevent">

<jpAttrMaps>

<jpAttrMap code="eventGuid" map="eventguid"/>

</jpAttrMaps>

</jpClassMap>

</jpClassMaps>

### Функция «Действия с объектами меты»

Модуль «jprime-common-starter» предоставляет API для взаимодействия с объектами через метамодель и кодовые имена классов/атрибутов.

API модуля представлено следующими методами:

* Чтение данных
* Создание данных
* Обновление данных
* Удаление данных
* Поиск данных

Метод поиска данных поддерживает следующие операторы поиска:

* Значение не указано;
* Значение указано;
* Значение равно указанному;
* Значение не равно указанному;
* Значение больше указанного;
* Значение больше или равно указанному;
* Значение меньше указанному;
* Значение меньше или равно указанному;
* Значение равно указанному году;
* Значение не равно указанному году;
* Значение больше указанного года;
* Значение больше или равно указанному года;
* Значение меньше указанного года;
* Значение меньше или равно указанного года;
* Значение равно указанному месяцу;
* Значение не равно указанному месяцу;
* Значение больше указанного месяца;
* Значение больше или равно указанному месяца;
* Значение меньше указанного месяца;
* Значение меньше или равно указанного месяца;
* Значение равно указанному дню;
* Значение не равно указанному дню;
* Значение больше указанного дня;
* Значение больше или равно указанному дню;
* Значение меньше указанного дня;
* Значение меньше или равно указанного дню;
* Значение находится в указанном диапазоне;
* Ссылочный атрибут содержит объекты по указанному условию;
* Ссылочный атрибут не содержит объекты по указанному условию;
* Значение содержит указанное;
* Нечеткий поиск;
* Нечеткий поиск с учетом порядка слов;
* Начинается с указанного;
* Не начинается с указанного;
* Значение находится в указанном списке;
* Значение(диапазон) содержит указанный диапазон;
* Значение(диапазон) содержит указанный элемент;
* Значение(диапазон) содержится в указанном диапазоне;
* Значение равно указанному диапазону;
* Значение не равно указанному диапазону;
* Значение больше указанного диапазона;
* Значение больше или равно указанному диапазону;
* Значение меньше указанного диапазона;
* Значение больше или равно указанному диапазону.

Возможность применения того или иного оператора поиска завит от типа атрибута.

### Функция «Управление доступом»

Модуль «jprime-common-starter» реализует возможность управления доступа к объектам как на основе ролей - Role-based access control (RBAC), так и на основе атрибутов - Attribute-based access control (ABAC).

#### Настройки RBAC

Настройки RBAC представляют собой разрешительные и запретительные настройки доступа для определенных ролей.

Структура настроек приведена в Таблица 9.

Таблица 9.Структура настроек RBAC

| Свойство | Описание |
| --- | --- |
| code | Уникальный код настроек доступа |
| access | Список настроек по ролям |
| type | Тип настройки: разрешительный или на запрет |
| role | Кодовое имя роли |
| read | Доступ на чтение |
| create | Доступ на создание |
| update | Доступ на изменение |
| delete | Доступ на удаление |

##### Способы описания настроек

Настройки доступа RBAC могут быть описаны через java-аннотацию или путем загрузки xml-файла.

###### Java-аннотация

Аннотация описания настроек доступа RBAC должна подчиняться структуре @JPPackages в приведенном ниже примере:

@JPPackages(

{

@JPPackage(

code = "commonAccess",

access = {

@JPAccess(

type = JPAccessType.PERMIT,

role = "AUTH\_ACCESS",

read = true,

create = true,

update = true,

delete = true

)

}

),

@JPPackage(

code = "commonDenied",

access = {

@JPAccess(

type = JPAccessType.PROHIBITION,

role = "AUTH\_ACCESS",

read = true,

create = true,

update = true,

delete = true

)

}

)

}

)

###### XML

Для загрузки описания настроек доступа RBAC через xml файлы должны подчиняться структуре, приведенной в примере ниже:

<?xml version="1.0" encoding="utf-8"?>

<jpSecurity>

<jpPackages>

<jpPackage code="commonAccess">

<jpPermitAccess>

<jpAccess read="true" create="true" update="true" delete="true" role="AUTH\_ACCESS"/>

</jpPermitAccess>

</jpPackage>

<jpPackage code="commonDenied">

<jpProhibitionAccess>

<jpAccess read="true" create="true" update="true" delete="true" role="AUTH\_ACCESS"/>

</jpProhibitionAccess>

</jpPackage>

</jpPackages>

</jpSecurity>

#### Настройки ABAC

Настройки ABAC реализуют разграничение доступа на основе атрибутов субъекта, окружения и ресурса.

Структура настроек ABAC приведена в Таблица 10.

Таблица 10.Структура настроек ABAC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа политик | | | | |
| Свойство | Описание | | | |
| name | Название группы политики | | | |
| qName | QName названия | | | |
| jpClasses | Список классов, к которым применяется группа политик. Если список пуст, то ко всем данным | | | |
| policies | Список политик | | | |
| Политика | | | | |
| Свойство | Описание | | | |
| name | Название политики | | | |
| qName | QName названия | | | |
| actions | Действия, к которым применяется политика (read, create, update, delete) | | | |
| subjectRules | Список правил для пользователя | | | |
| resourceRules | Список правил для объекта | | | |
| environmentRules | Список правил для окружения | | | |
| Правило для пользователя | | | | |
| Свойство | | Описание | | |
| name | | Название правила | | |
| qName | | QName правила | | |
| effect | | Возвращает тип доступа (разрешительный/запретительный) permit/prohibition | | |
| username | | Условие на логин | | |
| role | | Условие на роли | | |
| orgId | | Условие на организацию | | |
| depId | | Условие на подразделение | | |
| Правило для окружения | | | | |
| Свойство | | | Описание | |
| name | | | Название правила | |
| qName | | | QName правила | |
| effect | | | Возвращает тип доступа (разрешительный/запретительный) permit/prohibition | |
| ip | | | Условие на IP | |
| time | | | Условие на время применения политики | |
| time.daysOfWeek | | | Дни: пнд/вск | |
| time.fromTime | | | Начало периода в формате HH:mm | |
| time.toTime | | | Окончание периода в формате HH:mm | |
| time.fromDateTime | | | Начало периода в формате yyyy-MM-dd'T'HH:mm:ss | |
| time.toDateTime | | | Окончание периода в формате yyyy-MM-dd'T'HH:mm:ss | |
| Правило для объекта | | | | |
| Свойство | | | Описание | |
| name | | | Название правила | |
| qName | | | QName правила | |
| effect | | | Возвращает тип доступа (разрешительный/запретительный) permit/prohibition | |
| attr | | | Кодовое имя атрибута | |
| cond | | | Условие на атрибут | |
| Условие | | | | |
| Свойство | | | | Описание |
| in | | | | Список значений |
| notIn | | | | Список значений |

##### Способы описания настроек

Настройки доступа ABAC могут быть описаны через аннотацию или путем загрузки xml-файла.

###### Java-аннотация

Аннотация описания настроек доступа RBAC должна подчиняться структуре @ JPPolicySets в приведенном ниже примере:

@JPPolicySets(

value = {

@JPPolicySet(

name = "Группа политик annotation-тест",

qName = "jpPolicySet.annotationTest",

jpClasses = {"main"},

policies = {

@JPPolicy(

name = "Политика Создание",

qName = "jpPolicySet.annotationTest.create",

actions = {JPAction.CREATE},

subjectRules = {

@JPSubjectRule(

name = "Доступно только роли TEST\_ROLE",

qName = "jpPolicySet.annotationTest.create.rule1",

role = @JPCond(in = {"ADMIN\_ROLE", "TEST1\_ROLE"}),

effect = JPAccessType.PERMIT

)

},

environmentRules = {

@JPEnvironmentRule(

name = "Доступно только в течении для 2010-01-10",

qName = "jpPolicySet.annotationTest.create.envRule1",

time = @JPTime(

daysOfWeek = {DayOfWeek.MONDAY, DayOfWeek.THURSDAY},

fromTime = "10:00",

toTime = "17:59",

fromDateTime = "2010-01-10T00:00:00",

toDateTime = "2020-01-10T23:59:59"

)

)

}

),

@JPPolicy(

name = "Политика Обновление",

qName = "jpPolicySet.annotationTest.update1",

actions = {JPAction.READ, JPAction.UPDATE},

subjectRules = {

@JPSubjectRule(

name = "Доступно только роли ADMIN\_ROLE",

qName = "jpPolicySet.annotationTest.update1.rule1",

role = @JPCond(in = {"ADMIN\_ROLE"}),

effect = JPAccessType.PERMIT

)

}

),

@JPPolicy(

name = "Политика Обновление",

qName = "jpPolicySet.annotationTest.update2",

actions = {JPAction.READ, JPAction.UPDATE},

subjectRules = {

@JPSubjectRule(

name = "Доступно только роли TEST1\_ROLE",

qName = "jpPolicySet.annotationTest.update2.rule1",

role = @JPCond(in = {"TEST1\_ROLE"}),

effect = JPAccessType.PERMIT

)

},

resourceRules = {

@JPResourceRule(

name = "Доступны test1 и test2",

qName = "jpPolicySet.annotationTest.update2.rule1",

attr = "name",

cond = @JPCond(in = {"test1", "test2"}),

effect = JPAccessType.PERMIT

),

@JPResourceRule(

name = "Доступны все, кроме test2",

qName = "jpPolicySet.annotationTest.update2.rule2",

attr = "name",

cond = @JPCond(notIn = {"test2"}),

effect = JPAccessType.PERMIT

),

@JPResourceRule(

name = "Не доступен test3",

qName = "jpPolicySet.annotationTest.update3.rule3",

attr = "name",

cond = @JPCond(in = {"test3"}),

effect = JPAccessType.PROHIBITION

)

}

),

}

)

}

)

###### XML

Для загрузки описания настроек доступа ABAC через xml файлы должны подчиняться структуре, приведенной в примере ниже:

<?xml version="1.0" encoding="utf-8"?>

<jpAbac>

<jpPolicySets>

<jpPolicySet>

<name>Группа политик xml-тест</name>

<qName>jpPolicySet.xmlTest</qName>

<jpClasses>

<jpClass>test1</jpClass>

<jpClass>test2</jpClass>

</jpClasses>

<jpPolicies>

<jpPolicy>

<name>Политика Создание</name>

<qName>jpPolicySet.xmlTest.create</qName>

<actions>

<action>create</action>

</actions>

<subjectRules>

<subjectRule>

<name>Доступно только роли TEST\_ROLE</name>

<qName>jpPolicySet.xmlTest.create.rule1</qName>

<username/>

<role>

<in>

<value>TEST\_ROLE</value>

</in>

<notIn>

<value>TEST\_ROLE</value>

</notIn>

</role>

<orgId/>

<depId/>

<effect>permit</effect>

</subjectRule>

</subjectRules>

<resourceRules>

<resourceRule>

</resourceRule>

</resourceRules>

<environmentRules>

<environmentRule>

<name>Доступно в определенные часы</name>

<qName>jpPolicySet.xmlTest.create.environmentRule1</qName>

<effect>permit</effect>

<time>

<daysOfWeek>6,7</daysOfWeek>

<fromTime>10:00</fromTime>

<toTime>17:59</toTime>

<fromDateTime>2010-01-10T00:00:00</fromDateTime>

<toDateTime>2020-01-10T23:59:59</toDateTime>

</time>

</environmentRule>

</environmentRules>

</jpPolicy>

<jpPolicy>

<name>Политика Обновление</name>

<qName>jpPolicySet.xmlTest.update</qName>

<actions>

<action>read</action>

<action>update</action>

</actions>

<subjectRules>

<subjectRule>

</subjectRule>

</subjectRules>

<resourceRules>

<resourceRule>

<name>Доступно только автору</name>

<qName>jpPolicySet.xmlTest.update.rule1</qName>

<attr>userOwnerId</attr>

<cond>

<in>

<value>{AUTH\_USERID}</value>

</in>

</cond>

<effect>permit</effect>

</resourceRule>

</resourceRules>

<environmentRules>

<environmentRule>

</environmentRule>

</environmentRules>

</jpPolicy>

</jpPolicies>

</jpPolicySet>

</jpPolicySets>

</jpAbac>

### Функция «Валидация»

Перед осуществлением действия над объектами в Платформе реализована возможность проверки данных с помощью валидаторов. Например, валидность СНИЛС.

На один класс может быть настроено множество валидаторов, а один валидатор может применяться ко многим классам.

### Функция «Расчет значений по умолчанию»

Модуль «jprime-common-starter» предоставляет возможность рассчитывать значения по умолчанию при создании объекта.

На один класс может быть настроено множество расчетных блоков, а один расчет может применяться ко многим классам.

### Функция «Дополнение значений»

Модуль «jprime-common-starter» предоставляет возможность рассчитать значения атрибутов на основании имеющихся данных.

На один класс может быть настроено множество расчетных блоков, а один расчет может применяться ко многим классам.

### Функция «Определение дополнительных сведений по объекту»

Модуль предоставляет возможность возвратить набор информационных подсказок.

### Функция «Поддержка разных форматов JSON для хранения и отображения»

В модуле реализован интерфейс, который позволяет использовать разные форматы для хранения и отображения JSON.

### Функция «Поддержка получения данных через Java-код»

Источник данных через Java-код реализует интерфейс и применяется к объектам класса, указанных в аннотации @JPClassesLink.

@JPClassesLink(

jpClasses = {TestClass.CLASS\_CODE}

)

public class TestObjectAddInfo implements JPObjectAddInfo {

@Override

public Collection<AddInfo> getAddInfo(JPObjectAddInfoParams params) {

Collection<AddInfo> result = new ArrayList<>();

result.add(

AddInfoBean.newBuilder().code("test1").info("Данные корректны").build()

);

return result;

}

}

На один класс может быть настроено множество реализаций, данные которых в итоге складываются.

### Функция «Поддержка получения данных через SQL»

В Платформе реализовано определение дополнительных свойств через SQL-запрос. Подробнее см. модуль «jprime-dataaccess-jdbc-addinfos».

### Функция «Вызов обработчика CRUD-операций над объектами»

В модуле реализованы обработчики (хэндлеры) над CRUD-операциями, которые позволяют обработать следующие события:

* Перед созданием;
* Перед обновлением;
* Перед удалением;
* После создания;
* После обновления;
* После удаления.

### Функция «Работа с событиями»

Все события в системе являются ассинхронными и делятся на системные и пользовательские. Более подробное описание приведено в модуле «jprime-common-core».

#### Системные события

Системные события инициируются и обрабатываются программным кодом и используются для передачи информации между сервисами.

#### Пользовательские события

Пользовательское событие может быть инициировано в любом сервисе системы.

События инициируются программным кодом и используются для отображения информации конечному пользователю.

## Модуль «jprime-reactive-starter»

Модуль «jprime-reactive-starter» содержит типовую логику, API и базовые реализации реактивных функций системы.

## Модуль «jprime-dynamicmeta-core»

Модуль «jprime-dynamicmeta-core» предоставляет базовое API для работы c динамическим метаописанием. Динамическое метаописание подразумевает возможность создания и редактирования метаданных через подключаемый к платформе пользовательский интерфейс.

## Модуль «jprime-dynamicmeta-core-jdbc»

Модуль «jprime-dynamicmeta-core-jdbc» является реализацией функций работы с динамической метой, которая сопоставляется с БД через jdbc-драйвер.

## Модуль «jprime-dynamicmeta-editor-starter»

Модуль «jprime-dynamicmeta-editor-starter» предоставляет REST API для осуществления операций с динамической метой, такие как:

* создание меты с указанием маппинга;
* обновление меты с указанием маппинга;
* чтение меты с указанием маппинга;
* поиск меты;
* чтение определенной меты с указанием маппинга;
* удаление определенной меты с указанием маппинга;
* получение регистра значений;
* инициализация меты;
* синхронизация структуры метаописания и хранилища;
* синхронизация структуры метаописания и хранилища определенного класса;

Также модуль предоставляет возможность создавать мету по шаблонам и реализует следующие методы работы с шаблонами меты:

* создание шаблона меты;
* обновление шаблона меты;
* чтение шаблона меты;
* поиск шаблона меты;
* чтение определенного шаблона меты;
* удаление определенного шаблона меты с указанием маппинга;
* получение свойств типа атрибута;
* добавление обратной ссылки в уже имеющийся класс.

Кроме того, модуль обеспечивает операции с файловым хранилищем:

* получение списка файловых хранилищ;
* описание проверки имени хранилища файлов;
* проверка имени хранилища файлов.

## Модуль «jprime-dynamicmeta-editor-jdbc»

Модуль «jprime-dynamicmeta-editor-jdbc» является реализацией функций модуля «jprime-dynamicmeta-editor-starter» и обеспечивает хранение динамической меты в БД через jdbc-драйвер.

## Модуль «jprime-dynamicmeta-editor-imex»

Модуль «jprime-dynamicmeta-editor-imex» является опциональной реализацией функций модуля «jprime-dynamicmeta-editor-starter» и обеспечивает процессы экспорта и импорта настроек динамической меты, такие как:

* экспорт настроек меты;
* генерация и экспорт метаописания в xml-файл;
* генерация и экспорт метаописания в Java-аннотацию;
* импорт настроек меты.

## Модуль «jprime-dynamicmeta-loader-starter»

Модуль «jprime-dynamicmeta-loader-starter» API для загрузки метаописания на основании какого-либо хранилища.

## Модуль «jprime-dynamicmeta-loader-jdbc»

Модуль «jprime-dynamicmeta-loader-jdbc» является реализацией для загрузки метаописания из БД через jdbc-драйвер.

## Модуль «jprime-meta-rest»

Модуль Платформы «jprime-meta-rest» предоставляет REST API для публикации метаданных на чтение. Для внешнего вызова доступны следующие REST-методы:

### Получение типов атрибутов

GET /meta/v1/attrTypes/

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Утилита отработала успешно |

[

{

"code": "string",

"title": "Строка"

},

{

"code": "long",

"title": "Целочисленный (длинный)"

},

...

]

| Параметры | Описание |
| --- | --- |
| code | Код типа атрибута |
| title | Название типа атрибута |

### Получение всего списка метаописаний

GET /meta/v1/jpClasses/

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Утилита отработала успешно |

{

"totalCount": 4,

"classes": [

{

"guid": "cc53f898-ceec-49b3-81a3-43eb3fdc43f0",

"code": "userEvent",

"qName": "common.userEvent",

"description": "Таблица с информацией о пользовательских событиях",

"jpPackage": "adminAccess",

"immutable": true,

"attrs": [

{

"guid": "85e203ef-e611-4983-a102-fa046f45874c",

"code": "eventCode",

"qName": "common.userevent.eventCode",

"description": "Код события",

"type": "string"

},

...

]

},

...

]

}

| Параметр | Описание |
| --- | --- |
| totalCount | Общее количество классов |
| classes | Список метаописаний |

### Получение метаописания определенного класса

GET /meta/v1/jpClasses/<classCode>

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Утилита отработала успешно |

{

"guid": "cc53f898-ceec-49b3-81a3-43eb3fdc43f0",

"code": "userEvent",

"qName": "common.userEvent",

"description": "Таблица с информацией о пользовательских событиях",

"jpPackage": "adminAccess",

"immutable": true,

"attrs": [

{

"guid": "85e203ef-e611-4983-a102-fa046f45874c",

"code": "eventCode",

"qName": "common.userevent.eventCode",

"description": "Код события",

"type": "string"

},

{

"guid": "c57d1be4-33e6-4ef9-a87c-f6d23bb4b141",

"code": "removeDate",

"qName": "common.userevent.removeDate",

"description": "Дата, после которой событие будет удалено",

"type": "datetime"

},

{

"guid": "749adfcf-67be-4256-8abc-2a24b9568341",

"code": "fileName",

"qName": "fileName",

"name": "Название файла в хранилище",

"shortName": "Название файла в хранилище",

"description": "Название файла в хранилище",

"jpPackage": null,

"identifier": false,

"mandatory": false,

"type": "file",

"length": null,

"refJpClass": null,

"refJpAttr": null,

"refJpFile": {

"titleAttr": "fileTitle",

"extAttr": "fileExt",

"sizeAttr": "fileSize",

"dateAttr": "fileDate",

"infoAttr": "fileInfo"

}

},

{

"guid": "a717e338-1e08-4610-917f-9c9969db51b9",

"code": "jsonattr",

"qName": "common.userevent.jsonattr",

"name": "Test json attrs",

"shortName": "Test json attrs",

"description": "Test json attrs",

"jpPackage": null,

"identifier": false,

"mandatory": false,

"type": "json",

"length": null,

"refJpClass": null,

"refJpAttr": null,

"refJpFile": null,

"jpProps": [

{

"code": "testprop",

"qName": "common.userevent.jsonprop",

"name": "Test Prop",

"shortName": null,

"description": null,

"mandatory": false,

"multiple": false,

"type": "element",

"length": null,

"refJpClassCode": null,

"refJpAttrCode": null,

"jpProps": [

{

"code": "innertestprop",

"qName": "common.userevent.injsonprop",

"name": "Inner Test Prop",

"shortName": null,

"description": null,

"mandatory": false,

"multiple": false,

"type": "string",

"length": null,

"refJpClassCode": null,

"refJpAttrCode": null,

"jpProps": []

}

]

}

]

}

...

]

}

#### Описание параметров

Параметры класса

| Параметр | Описание |
| --- | --- |
| guid | Глобальный идентификатор класса |
| immutable | Признак неизменяемости метаописания |
| attrs | Список атрибутов класса |

Параметры атрибута

| Параметр | Описание |
| --- | --- |
| guid | Глобальный идентификатор |
| code | Кодовое имя атрибута |
| qName | Полный код атрибута |
| name | Название атрибута |
| shortName | Короткое название атрибута |
| description | Описание атрибута |
| jpPackage | Настройка доступа |
| identifier | Признак идентификатора |
| mandatory | Признак обязательности |
| type | Тип атрибута |
| updatable | Признак обновляемости значения атрибута |
| length | Длина (для строковых полей) |
| refJpClass | Кодовое имя класса, на который ссылается |
| refJpAttr | Кодовое имя атрибута ссылочного класса |
| refJpFile | Описание файлового атрибута |
| simpleFraction | Описание простой дроби |
| money | Настройки денежного типа |
| jpProps | Схема свойств псевдо-меты |

Параметры описания файлового атрибута (type:file)

| Параметр | Описание |
| --- | --- |
| titleAttr | Кодовое имя атрибута - Заголовок файла |
| extAttr | Кодовое имя атрибута - Расширение файла |
| sizeAttr | Кодовое имя атрибута - Размер файла |
| dateAttr | Кодовое имя атрибута - Дата файла |
| infoAttr | Кодовое имя атрибута - Ддополнительная информация о файле |

Параметры описания простой дроби (type:simpleFraction)

| Параметр | Описание |
| --- | --- |
| integerAttr | Атрибут для хранения - Целая часть дроби |
| denominatorAttr | Атрибут для хранения - Знаменатель дроби |

## Модуль «jprime-common-rest»

Модуль Платформы «jprime-common-rest» предоставляет REST API для работы с данными, на основе метаописания.

В модуле используются такие настройки, как:

* время ожидания запроса;
* проверка максимального количества (limit) в выборке;
* максимальное количество в выборке через API;
* признак добавления в ответ hateoas ссылок для ссылочных объектов.

Для внешнего вызова доступны следующие REST-методы работы с данными:

### Создание объекта (JSON)

POST /api/v1/<код метакласса>

* формат запроса

{

"classCode": "mpCompany",

"data": {

...

}

}

* формат ответа:

{

"id": "1",

"classCode": "mpCompany",

"data": {

...

}

}

### Обновление объекта (JSON)

PUT /api/v1/<код метакласса>

* формат запроса

{

"id": "1",

"classCode": "mpCompany",

"data": {

...

}

}

* формат ответа:

{

"id": "1",

"classCode": "mpCompany",

"data": {

...

}

}

### Удаление объекта

DELETE /api/v1/<код метакласса>/<идентификатор объекта>

### Создание объекта с файлом (через multipart form-data)

POST /api/v1/<код метакласса>

* формат запроса

| Свойство | Описание |
| --- | --- |
| body | Настройки создания в json |
| attr1 | Файл для атрибута attr1 (тип атрибута = файл) |
| attr2 | Файл для атрибута attr2 (тип атрибута = файл) |

* формат ответа:

{

"id": "1",

"classCode": "mpCompany",

"data": {

...

}

}

### Обновление объекта с файлом (через multipart form-data)

PUT /api/v1/<код метакласса>

* формат запроса

| Свойство | Описание | Content Type |
| --- | --- | --- |
| body | Настройки обновления в json | application/json;charset = UTF-8 |
| attr1 | Файл для атрибута attr1 (тип атрибута = файл) |  |
| attr2 | Файл для атрибута attr2 (тип атрибута = файл) |  |

* формат ответа:

{

"id": "1",

"classCode": "mpCompany",

"data": {

...

}

}

### Скачивание файла из файлового атрибута

GET /api/v1/{code}/{objectId}/file/{attrCode}/{bearer>}

| Свойство | Описание |
| --- | --- |
| code | код метакласса |
| objectId | идентификатор объекта |
| attrCode | кодовое имя атрибута типа файл |
| bearer | токен авторизации |

### Скачивание архива файлов из файлового атрибута по обратным ссылкам

GET /api/v1/{code}/search/file/{attrCode}/{linkCode}/{linkValue}/{bearer}

| Свойство | Описание |
| --- | --- |
| code | код метакласса |
| attrCode | кодовое имя атрибута типа файл |
| linkCode | кодовое имя атрибута, по которому строится обратная ссылка |
| linkValue | значения атрибута, по котрому строится обратная ссылка |
| bearer | токен авторизации |

### Получение списка объектов

GET /api/v1/<код метакласса>?offset=<>&limit=<>

* ответ

{

"offset": 0,

"limit": 10,

"objectsCount": 1,

"classCode": "mpCompany",

"objects": [

{

"id": 1800065421,

"classCode": "mpCompany",

"title": null,

"data": {

"ownerType": 0,

"dateReg": "2012-04-09T20:00:00.000+0000",

"educationType": null,

"municipality": null

}

}

]

}

### Осуществление поиска

Общий формат:

POST /api/v1/<код метакласса>/search

Формат с предзаполенным условием:

POST /api/v1/<код метакласса>/search/<код атрибута>/<значение>

Примеры:

* название содержит

{

"offset": 0,

"limit": 50,

"filter": {

"cond": {

"attr": "longName",

"like": "МАЯК"

}

}

}

* название пустое

{

"offset": 0,

"limit": 50,

"filter": {

"cond": {

"attr": "longName",

"isNull": true

}

}

}

* название не пустое

{

"offset": 0,

"limit": 50,

"filter": {

"cond": {

"attr": "longName",

"isNotNull": true

}

}

}

* Выбор организации, у которой есть адрес, название которого содержит "Девяткино" + подсчет точного кол-во таких объектов

{

"offset": 0,

"limit": 50,

"totalCount": true,

"filter": {

"and": [

{ "cond": {

"attr": "addresses",

"exists": {

"cond":{"attr": "fullAddress","like": "Девяткино"}

}

}

}

]

}

}

* название начинается ИЛИ инн = с сортировкой по указанным атрибутам

{

"limit": "10",

"offset": "0",

"filter": {"or": [

{"cond": {"attr":"longName", "startsWith": "ООО"}},

{"cond": {"attr":"inn", "eq": "7802199633"}}

]},

"orders": [{"asc":"inn"}, {"desc":"longName"}]

}

* объекты, удовлетворяющие указанному условию на указанную дату

{

"offset": 0,

"limit": 50,

"filter": {

"cond": {

"feature": "pers\_date\_n\_lt",

"checkDay": "2019-02-01"

}

}

}

### Расчет значений агрегаций

POST /api/v1/<код метакласса>/aggregate

– позволяет получить агрегационные значения для указанных атрибутов.

Возможные агрегации:

| Агрегация | Код | Тип атрибута: string | Тип атрибута: date | Тип атрибута: datetime/ timestamp | Тип атрибута: time | Тип атрибута: int, long, double, biginteger |
| --- | --- | --- | --- | --- | --- | --- |
| Максимум | MAX | + | + | + | - | + |
| Минимум | MIN | + | + | + | - | + |
| Среднее | AVG | - | - | - | - | + |
| Сумма | SUM | - | - | - | - | + |
| Количество | COUNT | + | + | + | + | + |
| Количество уникальных | COUNT\_DISTINCT | + | + | + | + | + |

**Формат запроса**

{

"aggrs": [

{

"alias": "a1",

"attr": "creationDate",

"operator": "MAX"

},

{

"alias": "a2",

"attr": "creationDate",

"operator": "MIN"

}

],

"filter": {

"cond": {

"attr": "creationDate", "gte": "2020-04-01T15:33:03.468+0300"

}

}

}

| Свойство | Описание |
| --- | --- |
| aggrs | Набор агрегаций |
| alias | Псеводним агрегации, под которым будет отображен результат |
| attr | Кодовое имя атрибута, на который накладывается агрегация |
| operator | Код агрегации |
| filter | Ограничение выборки |

**Формат ответа**

Ответ в aggrs является парой alias из первичного запроса - значение агрегации

{

"classCode": "mpCompany",

"aggrs": {

"a1": "2020-04-16T15:33:03.468+0300",

"a2": "2020-04-02T12:15:00.000+0300"

}

}

| Свойство | Описание |
| --- | --- |
| classCode | Кодовое имя класса |
| aggrs | Набор значений агрегаций |

### 11. Получение значения по умолчанию

POST /api/v1/<код метакласса>/defvalue

– позволяет получить значения по умолчанию перед созданием объекта

**Формат запроса**

{

"id": "1",

"classCode": "mpCompany",

"refAttrCode": "org",

"data": {

...

}

}

| Свойство | Описание |
| --- | --- |
| id | Идентификатор объекта, из которого создаем (может быть не указан) |
| classCode | Кодовое имя класса объекта, из которого создаем (может быть не указан) |
| data | Данные объекта, из которого создаем (может быть не указан) |
| refAttrCode | Атрибут текущего класса, по которому строилась ссылка (может быть не указан) |

**Формат ответа**

{

"classCode": "mpDoc",

"data": {

"userOwnerId": "9999999912",

"userEditorId": "9999999912",

"citizenship": 643,

"changeDate": "2020-04-23T18:25:23.240+0300",

"creationDate": "2020-04-23T18:25:23.240+0300",

"status": 1

}

}

| Свойство | Описание |
| --- | --- |
| classCode | Кодовое имя класса |
| data | Значения по умолчанию для атрибутов |

### 12. Дополнение значений

POST /api/v1/<код метакласса>/applyvalue

– позволяет дополнить значения на основании имеющихся. В ответе будут только данные, подлежащие изменению, входящая data в ответе не дублируется.

**Формат запроса**

{

"id": 15,

"classCode": "universityNeat",

"attrs": ["a1"],

"data": {

"a1":1,

"a2":2

}

}

| Свойство | Описание |
| --- | --- |
| id | Идентификатор объекта |
| classCode | Кодовое имя класса объекта |
| attrs | Список атрибутов, значения которых изменились |
| data | Данные объекта |

**Формат ответа**

{

"id": 15,

"classCode": "universityNeat",

"data": {

"phone": "911"

}

}

| Свойство | Описание |
| --- | --- |
| id | Идентификатор объекта |
| classCode | Кодовое имя класса объекта |
| data | Новые данные объекта |

### 13. Получение дополнительной информации по объекту

POST /api/v1/<код метакласса>/addinfo

– позволяет получить текстовое описание об объекте.

**Формат запроса**

{

"id": "10"

}

| Свойство | Описание |
| --- | --- |
| id | Идентификатор объекта |

**Формат ответа**

[

{

"code": "test1",

"info": "Данные корректны"

},

...

]

| Свойство | Описание |
| --- | --- |
| code | Код сообщения |
| info | Текст сообщения |

## Модуль «jprime-jpobjecthistory-storage-starter»

Модуль «jprime-jpobjecthistory-storage-starter» предоставляет программный интерфейс для сохранения истории изменения объекта.

## Модуль «jprime-jpobjecthistory-storage-jdbc»

Модуль «jprime-jpobjecthistory-storage-jdbc» отвечает за реализацию сохранения истории изменения объекта в БД через jdbc-драйвер.

Принцип работы модуля заключается в слушании топика Kafka на предмет событий создания/изменения/удаления объектов. Все полученные события автоматически сохраняются в определенной таблице.

## Модуль «jprime-jpobjecthistory-storage-rest»

Модуль отвечает за предоставление доступа к истории изменения объекта с помощью REST-методов.

Реализовано два метода получения списка истории объекта:

* Метод получения списка истории объектов № 1

– возвращает первые 50 записей из истории указанного объекта.

Общие параметры ответа:

| Параметр | Описание |
| --- | --- |
| offset | Начальное смещение выборки |
| limit | Кол-во объектов в выборке |
| classCode | Кодовое имя класса объекта |
| id | Идентификатор объекта |
| totalCount | Всего записей истории по этому объекту |
| changesCount | Количество объектов в выборке |
| changes | Список истории изменения объекта |

Вложенные параметры истории изменения:

| Параметр | Описание |
| --- | --- |
| act | Действие над объектом (create,update,delete) |
| classCode | Кодовое имя класса объекта |
| id | Идентификатор объекта |
| data | Вносимые изменения |
| date | Дата изменений |
| username | Имя пользователя |
| userIp | IP пользователя |

* Метод получения списка истории объектов № 2

– позволяет получить историю изменения по массиву объектов, задавая смещение и количество объектов выборке.

Параметры запроса:

| Параметр | Описание |
| --- | --- |
| offset | Начальное смещение выборки |
| limit | Кол-во объектов в выборке |
| classCode | Кодовое имя класса объекта |
| id | Идентификатор объекта |

Параметры ответа:

| Параметр | Описание |
| --- | --- |
| offset | Начальное смещение выборки |
| limit | Кол-во объектов в выборке |
| classCode | Кодовое имя к{}}ласса объекта |
| id | Идентификатор объекта |
| totalCount | Всего записей истории по этому объекту |
| changesCount | Количество объектов в выборке |
| changes | Список истории изменения объекта |

Вложенные параметры истории изменения:

| Параметр | Описание |
| --- | --- |
| act | Действие над объектом (create,update,delete) |
| classCode | Кодовое имя класса объекта |
| id | Идентификатор объекта |
| data | Вносимые изменения |
| date | Дата изменений |
| username | Имя пользователя |
| userIp | IP пользователя |

## Модуль «jprime-signer-starter»

Модуль «jprime-signer-starter» предназначен для сбора параметров для клиентской цифровой подписи и проверки валидности поступающих подписей.

## Модуль «jprime-signer-rest»

Модуль «jprime-signer-rest» предоставляет REST API для работы с цифровой подписью данных:

* получение pkcs7 хеша файла;
* получение pkcs7 хеша файла со штампом;
* сохранение подписи файла;
* проверка открепленной подписи файла;
* проверка прикрепленной/встроенной подписи файла.

## Модуль «jprime-signer-stamped-starter»

Модуль «jprime-signer-stamped-starter» реализует логику обработки запроса на постановку штампа подписи на файл.

Для корректной работы необходимо подключить одну из реализаций PdfConverter:

* Модуль «jprime-file-pfdconverter-local» - для локальной конвертации;
* Модуль «jprime-file-pfdconverter-jpremote» - для конвертации в микросервисной архитектуре.

После получения информации о файле из метаданных модуль обращается к сервису, реализованному в модуле «jprime-signer-core» (см. Модуль криптографии (crypto)).

# Пакет модулей авторизации «JPrime-auth»

Одной из особенностей платформы JPrime является отсутствие хранения какого-либо состояния процесса на сервере, что позволяет не только экономить ресурсы, но и балансировать нагрузку, не привязываясь к определенным ресурсам.

В JPrime за процесс авторизации и аутентификации отвечает отдельный сервис, работающий по протоколу Oauth 2.0. В задачи этого центра авторизации входит аутентификация пользователя, создание JWT-токена и раздача открытых ключей остальным сервисам для проверки JWT-токенов, передаваемых ими в запросах.

JWT-токен, помимо аутентификационной информации, может содержать любую другую, необходимую для бизнес-логики. Например, штатную структуру, адрес.

Пакет модулей авторизации в JPrime состоит из двух модулей, в задачи которых входит получение открытого ключа, расшифровка и проверка полученного JWT-токена. Состав пакета представлен в Таблица 11:

Таблица 11. Модули в составе с модуля авторизации

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-auth-jwt-starter | Базовый модуль авторизации JWT |
| jprime-auth-jwt | Типовой модуль авторизации JWT |

## Модуль «jprime-auth-jwt-starter»

Модуль «jprime-auth-jwt-starter» является базовым и содержит логику по работе с JWT. Реализация основана на Java JWT. При создании токена используются открытый и закрытые ключи по алгоритму RS256.

Открытый ключ передается в заголовке для валидации токена. Может быть выбрана одна из двух реализаций публикации открытого ключа: через REST метод или через Apache Kafka.

Закрытый ключ хранится в центре авторизации.

## Модуль «jprime-auth-jwt»

Модуль «jprime-auth-jwt» подключается к модулю «jprime-auth-jwt- starter» и расширяет его логику работы с авторизацией в части получения из токена информации о пользователе, которую можно использовать для прикладных задач.

# Пакет модулей поддержки криптографических функций «JPrime-crypto»

Платформа JPrime способна работать с цифровой подписью, а также с зашифрованными данными с помощью набора модулей поддержки криптографических функций. Модули поддержки криптографических функций используются для работы с электронной подписью, а также обеспечивают возможность наложения штампа с информацией о подписи на файл.

Перечень и описание модулей криптографии представлено в Таблица 12:

Таблица 12. Модули в составе модуля криптографии

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-signer-core | Модуль цифровой подписи |
| jprime-signer-client | Модуль клиентской обработки функций подписи |
| jprime-signer-stamped-core | Модуль наложения штампа с информацией о подписи на файл |
| jprime-cipher-core | Содержит логику по шифрации/дешифрации данных |

## Модуль «jprime-signer-core»

Модуль «jprime-signer-core» является основным для работы с электронной подписью. Модуль реализует сервис, который предназначен как для проверки открепленной подписи, так и для проверки прикрепленной/встроенной подписи в формате Base64.

Открепленная подпись представляет собой файл с подписью, который передается вместе с исходным подписанным файлом.

Прикрепленная подпись представляет собой один файл, в котором находится как подпись, так и контент исходного подписанного файла. Для просмотра подписанного контента необходим специальный софт, умеющий извлекать его.

Также в модуле есть логика подготовки данных для подписи.

## Модуль «jprime-signer-client»

Модуль «jprime-signer-client» предоставляет API для подключения по TCP-протоколу к внешнему сервису подписи, а также хэширования и валидации.

## Модуль «jprime-signer-stamped-core»

Модуль «jprime-signer-stamped-core» предназначен для наложения на файл штампа с информацией о подписи.

## Модуль «jprime-cipher-core»

Модуль «jprime-cipher-core» содержит логику по шифрации/дешифрации данных.

# Пакет модулей для работы с хранилищами «JPrime-storages»

В платформе JPrime реализована возможность интеграции с внешними системами и хранилищами данных с помощью пакета модулей для работы с хранилищами «JPrime-storages», который состоит из модулей, представленных в Таблица 13:

Таблица 13. Модули в составе модуля для работы с хранилищами

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-dataaccess-core | Базовое API для работы с хранилищами |
| jprime-dataaccess-jdbc | Модуль для подключения к любой СУБД через jdbc-драйвер |
| jprime-dataaccess-jdbc-settings-rest | REST API для публикация настроек и запросов для работы с БД |
| jprime-dataaccess-jdbc-addinfos | Модуль работы с дополнительными сведениями об объекте через jdbc-драйвер |
| jprime-dataaccess-jdbc-addinfos-rest | REST API для работы с дополнительными сведениями об объекте |
| jprime-dataaccess-r2dbc | Модуль для подключения к любой СУБД через r2dbc-драйвер |
| jprime-dataaccess-jpremote | Модуль работы с удаленным сервисами JPrime по REST |
| jprime-dataaccess-generated | Модуль работы с произвольными структурами данных |

## Модуль «jprime-dataaccess-core»

Модуль «jprime-dataaccess-core» предоставляет базовое API для работы с хранилищами, вводит базовые понятия. Такой подход позволяет разграничить уровень метаописания и хранилища данных. Прикладная логика решений взаимодействует с метаописанием, а в мете в свою очередь указывается код хранилища, которое может представлять из себя классическую базу данных, либо другой сервис, который берет данные где-то ещё, либо реактивную базу данных.

## Модуль «jprime-dataaccess-jdbc»

Модуль «jprime-dataaccess-jdbc» подключается к «jprime-dataaccess-core» и позволяет подключаться к любой СУБД через jdbc-драйвер. Он предназначен для обработки запросов и получения данных из БД, а также содержит основной блок функций, необходимых для работы с СУБД:

* Получение хранилища по коду;
* Проверка наличия таблицы в хранилище;
* Проверка наличия колонки в таблице;
* Получение шаблонного класса JdbcTemplate для прямого построения запроса;
* Построение сложных запросов (используется API QueryDsl);
* Массовая вставка/обновление объектов;
* Мониторинг времени выполнения долгих запросов;

В конфигурационном файле задаются необходимые настройки подключения к БД. Реализована возможность на каждое хранилище создать отдельный пул соединений.

## Модуль «jprime-dataaccess-jdbc-settings-rest»

Модуль «jprime-dataaccess-jdbc-settings-rest» предоставляет REST API для публикации настроек и запросов для работы с БД.

## Модуль «jprime-dataaccess-jdbc-addinfos»

Модуль работы с дополнительными сведениями об объекте через jdbc-драйвер.

## Модуль «jprime-dataaccess-jdbc-addinfos-rest»

Модуль «jprime-dataaccess-jdbc-addinfos-rest» предоставляет REST API для работы с дополнительными сведениями об объекте

Для внешнего вызова доступны следующие REST-методы:

### Получение списка дополнительной информации

GET /api/v1/{code}/addInfos/

* ответ

{

"totalCount": 3,

"count": 3,

"classCode": "addinfosSqlsetting",

"objects": [

{

"code": "f63cd37b-088d-4611-bf4f-34ae489e8a52",

"classCode": "suContracts",

"storageCode": "main",

"sql": "select guid, num from su\_contract where ouid = 1;",

"userEditorId": 62,

"userOwnerId": 62,

"changeDate": "2020-11-17T19:31:15.703+0300",

"creationDate": "2020-11-17T19:31:15.703+0300",

"version": "MjAyMi0wMS0xMFQxMzowNzoyNC4yNTI="

},

...

]

}

### Поиск дополнительной информации

POST /api/v1/{code}/addInfos/search/

* запрос

"offset": 1,

"limit": 50

* ответ

{

"totalCount": 3,

"count": 1,

"classCode": "addinfosSqlsetting",

"objects": [

{

"code": "f63cd37b-088d-4611-bf4f-34ae489e8a52",

"classCode": "suContracts",

"storageCode": "main",

"sql": "select guid, num from su\_contract where ouid = 1;",

"userEditorId": 62,

"userOwnerId": 62,

"changeDate": "2020-11-17T19:31:15.703+0300",

"creationDate": "2020-11-17T19:31:15.703+0300",

"version": "MjAyMi0wMS0xMFQxMzowNzoyNC4yNTI="

}

]

}

### Получение дополнительной информации по коду

GET /api/v1/{code}/addInfos/{objectId}/

* ответ

{

"code": "827d9db2-a57b-4da5-a8ae-1b62eb6128a5",

"classCode": "suContract",

"storageCode": "main",

"sql": "select guid, num from su\_contract where ouid = 1;;;",

"userEditorId": 62,

"userOwnerId": 62,

"changeDate": "2020-11-18T00:25:29.328+0300",

"creationDate": "2020-11-17T21:10:06.126+0300",

"version": "MjAyMi0wMS0xMFQxMzowNzoyNC4yNTI="

}

### Удаление дополнительной информации по коду

DELETE /api/v1/{code}/addInfos/{objectId}/

* ответ - статус 202

### Создание дополнительной информации по коду

POST /api/v1/{code}/addInfos/

* запрос

{

"storageCode":"main",

"sql":"select guid, num from su\_contract where ouid = 1;"

}

* ответ

{

"code": "827d9db2-a57b-4da5-a8ae-1b62eb6128a5",

"classCode": "suContract",

"storageCode": "main",

"sql": "select guid, num from su\_contract where ouid = 1;;;",

"userEditorId": 62,

"userOwnerId": 62,

"changeDate": "2020-11-18T00:25:29.328+0300",

"creationDate": "2020-11-17T21:10:06.126+0300",

"version": "MjAyMi0wMS0xMFQxMzowNzoyNC4yNTI="

}

### Обновление дополнительной информации по коду

PUT /api/v1/{code}/addInfos/

* запрос

{

"storageCode":"main",

"sql":"select guid, num from su\_contract where ouid = 1;",

"code":"827d9db2-a57b-4da5-a8ae-1b62eb6128a5",

"version": "MjAyMi0wMS0xMFQxMzowNzoyNC4yNTI="

}

* ответ

{

"code": "827d9db2-a57b-4da5-a8ae-1b62eb6128a5",

"classCode": "suContract",

"storageCode": "main",

"sql": "select guid, num from su\_contract where ouid = 1;;;",

"userEditorId": 62,

"userOwnerId": 62,

"changeDate": "2020-11-18T00:25:29.328+0300",

"creationDate": "2020-11-17T21:10:06.126+0300",

"version": "MjAyMi0wMS0xMFQxMzowNzoyNC4yNTI=",

"updatedUserDescription": "Иванов Владимир"

}

## Модуль «jprime-dataaccess-r2dbc»

Модуль «jprime-dataaccess-r2dbc» подключается к «jprime-dataaccess-jdbc-core» и позволяет подключаться к любой СУБД через r2db-драйвер.

## Модуль «jprime-dataaccess-jpremote»

Модуль «jprime-dataaccess-jpremote» предназначен для работы с удаленным сервисами по REST, при необходимости подключается к «jprime-dataaccess-jdbc-core» и «jprime-common-rest».

## Модуль «jprime-dataaccess-generated»

Модуль «jprime-dataaccess-generated» является реализацией для работы с произвольными структурами данных.

# Пакет модулей безопасности «JPrime-security»

Пакет модулей безопасности «JPrime-security» реализует возможности работы с динамическими настройками доступа к объектам как на основе ролей - Role-based access control (RBAC), так и на основе атрибутов - Attribute-based access control (ABAC), а также содержит REST-модуль для внешнего информирования об имеющихся настройках. Пакет состоит из модулей, представленных в Таблица 14.

Таблица 14. Модули в составе модуля безопасности

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-dynamicsecurity-core | Базовый модуль работы с динамическими настройками безопасности RBAC |
| jprime-dynamicsecurity-core-jdbc | Базовый модуль работы с динамическими настройками безопасности RBAC **в БД** |
| jprime-dynamicsecurity-loader-starter | API для загрузчика настроек доступа RBAC |
| jprime-dynamicsecurity-loader-jdbc | Реализация загрузчика настроек доступа RBAC из БД |
| jprime-dynamicsecurity-editor-starter | API и REST-методы редактирования настроек доступа RBAC |
| jprime-dynamicsecurity-editor-jdbc | Редактирования настроек доступа RBAC, хранящихся в БД |
| jprime-dynamicsecurity-editor-imex | Утилиты экспорт/импорт настроек доступа RBAC |
|  |  |
| jprime-dynamicsecurity-abac-core | Базовый модуль работы с динамическими настройками безопасности ABAC |
| jprime-dynamicsecurity-abac-core-jdbc | Базовый модуль работы с динамическими настройками безопасности ABAC **в БД** |
| jprime-dynamicsecurity-abac-loader-starter | API для загрузчика настроек доступа ABAC |
| jprime-dynamicsecurity-abac-loader-jdbc | Реализация загрузчика настроек доступа ABAC из БД |
| jprime-dynamicsecurity-abac-editor-starter | API и REST-методы редактирования настроек доступа ABAC |
| jprime-dynamicsecurity-abac-editor-jdbc | Редактирования настроек доступа ABAC, хранящихся в БД |
| jprime-dynamicsecurity-abac-editor-imex | Утилиты экспорт/импорт настроек доступа ABAC |
| jprime-security-rest | Модуль для REST-методов публикации настроек доступа |

## Модуль «jprime-dynamicsecurity-core»

Базовый модуль работы с динамическими настройками безопасности RBAC.

## Модуль «jprime-dynamicsecurity-core-jdbc»

Базовый модуль работы с динамическими настройками безопасности RBAC в БД через jdbc-драйвер.

## Модуль «jprime-dynamicsecurity-loader-starter»

Модуль «jprime-dynamicsecurity-loader-starter» содержит API для загрузчика настроек доступа RBAC

## Модуль «jprime-dynamicsecurity-loader-jdbc»

Модуль «jprime-dynamicsecurity-loader-jdbc» реализует загрузку настроек доступа RBAC из БД через jdbc-драйвер. Настройки доступа хранятся в хранилище с определенным кодом.

## Модуль «jprime-dynamicsecurity-editor-starter»

Модуль «jprime-dynamicsecurity-editor-starter» реализует API и REST-методы операций с настройками доступа RBAC, такие как:

* Создание настроек;
* Обновление настроек;
* Чтение настроек;
* Чтение определенных настроек;
* Удаление определенных настроек;
* Инициализация настроек доступа;
* Инициализация настроек доступа;
* Инициализация настроек доступа пользователя.

## Модуль «jprime-dynamicsecurity-editor-jdbc»

Модуль «jprime-dynamicsecurity-editor-jdbc» предназначен для редактирования настроек доступа RBAC, хранящихся в БД, через jdbc-драйвер.

## Модуль «jprime-dynamicsecurity-editor-imex»

Импорт и экспорт настроек доступа RBAC осуществляется посредством утилит, реализованных в модуле «jprime-dynamicsecurity-editor-imex».

Экспорт настроек доступа RBAC реализован методом POST и работает следующим образом:

* Формируется файл с данным экспортируемых настроек доступа
* Файл сохраняется в ФС-хранилище с определенным кодом
* Метаописание файла сохраняется в мета-хранилище также с определенным кодом
* Формируется пользовательское уведомление со ссылкой на сформированный файл

Импорт настроек доступа RBAC реализован методом POST и работает следующим образом:

* Файл сохраняется в ФС-хранилище с определенным кодом
* Метаописание файла сохраняется в мета-хранилище также с определенным кодом
* Настройки из файла переносятся в систему
* Формируется пользовательское уведомление со ссылкой на сформированный файл

## Модуль «jprime-dynamicsecurity-abac-core»

Базовый модуль работы с динамическими настройками безопасности ABAC.

## Модуль «jprime-dynamicsecurity-abac-core-jdbc»

Базовый модуль работы с динамическими настройками безопасности ABAC в БД через jdbc-драйвер.

## Модуль «jprime-dynamicsecurity-abac-loader-starter»

Модуль «jprime-dynamicsecurity-abac-loader-starter» содержит базовый API для загрузчика настроек доступа ABAC.

## Модуль «jprime-dynamicsecurity-abac-loader-jdbc»

Модуль «jprime-dynamicsecurity-abac-loader-jdbc» реализует загрузку настроек доступа ABAC из БД через jdbc-драйвер. Настройки доступа хранятся в хранилище с определенным кодом.

## Модуль «jprime-dynamicsecurity-abac-editor-starter»

Модуль «jprime-dynamicsecurity-abac-editor-starter» содержит базовый API и REST-методы редактирования настроек доступа ABAC, такие как:

* Создание настроек;
* Обновление настроек;
* Удаление определенных настроек;
* Инициализация настроек доступа:
* по идентификаторам,
* по пользователю;
* Полный список настроек политик доступа ABAC;
* Настройка политики доступа ABAC по коду;
* Поиск политик доступа ABAC.

## Модуль «jprime-dynamicsecurity-abac-editor-jdbc»

Модуль «jprime-dynamicsecurity-abac-editor-jdbc» предназначен для редактирования настроек доступа ABAC, хранящихся в БД.

## Модуль «jprime-dynamicsecurity-abac-editor-imex»

Импорт и экспорт настроек доступа ABAC осуществляется посредством утилит, реализованных в модуле «jprime-dynamicsecurity-abac-editor-imex».

Экспорт настроек доступа ABAC реализован методом POST и работает следующим образом:

* Формируется файл с данным экспортируемых настроек доступа
* Файл сохраняется в ФС-хранилище с определенным кодом
* Метаописание файла сохраняется в мета-хранилище также с определенным кодом
* Формируется пользовательское уведомление со ссылкой на сформированный файл

Импорт настроек доступа ABAC реализован методом POST и работает следующим образом:

* Файл сохраняется в ФС-хранилище с определенным кодом
* Метаописание файла сохраняется в мета-хранилище также с определенным кодом
* Настройки из файла переносятся в систему
* Формируется пользовательское уведомление со ссылкой на сформированный файл

## Модуль «jprime-security-rest»

Модуль «jprime-security-rest» предоставляет REST API для публикации рабочих настроек доступа на чтение как к статическим настройкам, так и проинициализированным динамическим.

Для внешнего вызова доступны следующие REST-методы:

### Получение всего списка настроек доступа

GET /access/v1/jpPackages/

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Утилита отработала успешно |

**для роли AUTH\_ADMIN**

[

{

"code": "adminAccess",

"name": "Полный доступ и только для роли ADMIN",

"description": "Полный доступ и только для роли ADMIN",

"qName": "",

"accesses": [

{

"type": "permit",

"role": "ADMIN",

"read": true,

"create": true,

"update": true,

"delete": true

}

]

},

...

]

для остальных ролей

[

{

"code": "adminAccess",

"name": "Полный доступ и только для роли ADMIN",

"description": "Полный доступ и только для роли ADMIN",

"qName": ""

},

...

]

| Параметр | Описание |
| --- | --- |
| totalCount | Общее количество классов |
| classes | Список метаописаний |

### Получение настроек доступа по указанному коду

GET /access/v1/jpPackages/<сode>

* роль AUTH\_ACCESS

| Параметр | Описание |
| --- | --- |
| сode | Код настройки. |

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Утилита отработала успешно |

для роли AUTH\_ADMIN

{

"code": "commonReadonly",

"name": "Доступ полный для роли ADMIN, остальным на чтение",

"description": "Доступ полный для роли ADMIN, остальным на чтение",

"qName": "",

"accesses": [

{

"type": "permit",

"role": "ADMIN",

"read": true,

"create": true,

"update": true,

"delete": true

},

{

"type": "permit",

"role": "AUTH\_ACCESS",

"read": true,

"create": false,

"update": false,

"delete": false

},

{

"type": "prohibition",

"role": "DISABLED",

"read": true,

"create": true,

"update": true,

"delete": true

}

]

}

**для остальных ролей**

{

"code": "commonReadonly",

"name": "Доступ полный для роли ADMIN, остальным на чтение",

"description": "Доступ полный для роли ADMIN, остальным на чтение",

"qName": ""

}

### Получение полного списка настроек доступа ко всем классам

GET /access/v1/jpClasses

* роль AUTH\_ACCESS
* ответ

[

{

"classCode": "<classCode1>",

"read": true,

"create": true,

"update": true,

"delete": true,

"editAttrs": {

"<attrCode1>": true,

"<attrCode2>": true,

"<attrCode3>": false

}

},

{

"classCode": "<classCode2>",

"read": true,

"create": true,

"update": true,

"delete": false,

"editAttrs": {

"<attrCode1>": true,

"<attrCode2>": true,

"<attrCode3>": false

}

}

]

| Параметр | Описание |
| --- | --- |
| classCode | Кодовое имя класса |
| read | Признак доступа на чтение объектов класса |
| create | Признак доступа на создание объектов класса |
| update | Признак доступа на обновление объектов класса |
| delete | Признак доступа на удаление объектов класса |
| editAttrs | Список атрибутов, доступных для редактирования |

Список атрибутов, доступных для редактирования содержит кодовое имя атрибута и признак возможности редактирования. Если кодового имени атрибута в editAttrs нет, то просмотр атрибута запрещен

### Получение настроек доступа к определенному классу

GET /access/v1/jpClasses/<classCode>

* роль AUTH\_ACCESS
* ответ

{

"classCode": "<classCode>",

"read": true,

"create": true,

"update": true,

"delete": true,

"editAttrs": {

"<attrCode1>": true,

"<attrCode2>": true,

"<attrCode3>": false

}

}

| Параметр | Описание |
| --- | --- |
| classCode | Кодовое имя класса |
| read | Признак доступа на чтение объектов класса |
| create | Признак доступа на создание объектов класса |
| update | Признак доступа на обновление объектов класса |
| delete | Признак доступа на удаление объектов класса |
| editAttrs | Список атрибутов, доступных для редактирования |

Список атрибутов, доступных для редактирования содержит кодовое имя атрибута и признак возможности редактирования. Если кодового имени атрибута в editAttrs нет, то просмотр атрибута запрещен

### Получение настроек доступа к определенному классу при указанном значении

GET /access/v1/jpClasses/<classCode>/<код атрибута>/<значение>

– используется для проверки доступа (обычно создания) по обратной ссылке.

* роль AUTH\_ACCESS
* ответ

{

"classCode": "<classCode>",

"read": true,

"create": true,

"update": true,

"delete": true,

"editAttrs": {

"<attrCode1>": true,

"<attrCode2>": true,

"<attrCode3>": false

}

}

| Параметр | Описание |
| --- | --- |
| classCode | Кодовое имя класса |
| read | Признак доступа на чтение объектов класса |
| create | Признак доступа на создание объектов класса |
| update | Признак доступа на обновление объектов класса |
| delete | Признак доступа на удаление объектов класса |
| editAttrs | Список атрибутов, доступных для редактирования |

Список атрибутов, доступных для редактирования содержит кодовое имя атрибута и признак возможности редактирования. Если кодового имени атрибута в editAttrs нет, то просмотр атрибута запрещен.

### Получение настройки доступа к определенному объекту

GET /access/v1/jpObjects/<classCode>/<objectId>

* роль AUTH\_ACCESS
* ответ

{

"objectClassCode": "<classCode>",

"objectId": "<objectId>",

"read": true,

"create": false,

"update": false,

"delete": false,

"editAttrs": {

"<attrCode1>": true,

"<attrCode2>": true,

"<attrCode3>": false

}

}

| Параметр | Описание |
| --- | --- |
| objectClassCode | Кодовое имя класса объекта |
| objectId | Идентификатор объекта |
| read | Признак доступа на чтение объекта |
| create | Признак доступа на создание объекта |
| update | Признак доступа на обновление объекта |
| delete | Признак доступа на удаление объекта |
| editAttrs | Список атрибутов, доступных для редактирования |

Список атрибутов, доступных для редактирования содержит кодовое имя атрибута и признак возможности редактирования. Если кодового имени атрибута в editAttrs нет, то просмотр атрибута запрещен

### Получение полного списка настроек политик доступа ABAC

GET /access/v1/policySets

* роль AUTH\_ADMIN
* ответ

[

{

"code": "8ffc9a11-6b78-44c5-b940-5d6bc4319e65",

"name": "Группа политик esrnOrgDepartment",

"qName": "jpPolicySet.esrnOrgDepartment",

"target": {

"jpClasses": [

"esrnOrgDepartment"

]

},

"policies": [

...

]

### Настройка политики доступа ABAC по коду

GET /access/v1/policySets/<код политики доступа>

* роль AUTH\_ADMIN
* ответ

{

"code": "8ffc9a11-6b78-44c5-b940-5d6bc4319e65",

"name": "Группа политик esrnOrgDepartment",

"qName": "jpPolicySet.esrnOrgDepartment",

"target": {

"jpClasses": [

"esrnOrgDepartment"

]

},

"policies": [

...

}

### Поиск политик доступа ABAC

POST /access/v1/policySets/search

– возвращает только политики, соответствующие запросу

* роль AUTH\_ADMIN
* запрос

{

"name": null,

"roles": ["AUTH\_ACCESS"],

"jpClassCodes": [ "esrnOrgDepartment"],

"useCreate": null,

"useRead": null,

"useUpdate": null,

"useDelete": null,

"usePermit": null,

"useProhibition": true,

"useEnviromentRules": null

}

| Параметр | Описание |
| --- | --- |
| name | Значение для поиска по имени |
| roles | Список ролей для фильтрации |
| jpClassCodes | Список классов для фильтрации |
| useCreate | Поиск по созданию |
| useRead | Поиск по чтению |
| useUpdate | Поиск по изменению |
| useDelete | Поиск по удалению |
| usePermit | Поиск по настройкам на разрешение |
| useProhibition | Поиск по настройкам на запрет |
| useEnviromentRules | Поиск по наличию правил окружения |

* ответ

[

{

"code": "8ffc9a11-6b78-44c5-b940-5d6bc4319e65",

"name": "Группа политик esrnOrgDepartment",

"qName": "jpPolicySet.esrnOrgDepartment",

"target": {

"jpClasses": [

"esrnOrgDepartment"

]

},

"policies": [

...

]

}

]

### Массовое получение настроек доступа к определенным объектам

POST /access/v1/policySets/search

* роль AUTH\_ACCESS
* запрос:

{

"ids": [

{

"objectClassCode": "studentsNeat",

"objectIds": [

"56",

"57"

]

},

...

]

}

* ответ:

{

"accessList": [

{

"objectClassCode": <classCode>,

"objectId": <objectId>,

"read": true,

"create": false,

"update": false,

"delete": false,

"editAttrs": {

"<attrCode1>": true,

"<attrCode2>": true,

"<attrCode3>": false

},

...

}

]

}

| Параметр | Описание |
| --- | --- |
| accessList | Список настроек доступа |
| objectClassCode | Кодовое имя класса объекта |
| objectId | Идентификатор объекта |
| read | Признак доступа на чтение объекта |
| create | Признак доступа на создание объекта |
| update | Признак доступа на обновление объекта |
| delete | Признак доступа на удаление объекта |
| editAttrs | Список атрибутов, доступных для редактирования |

Список атрибутов, доступных для редактирования содержит кодовое имя атрибута и признак возможности редактирования. Если кодового имени атрибута в editAttrs нет, то просмотр атрибута запрещен.

# Пакет модулей для работы с файлами «JPrime-files»

Платформа JPrime способна работать с файловой системой, выполняя базовые операции по конвертации файлов разных форматов. Функции работы с файловой системой и файловым хранилищем реализованы с помощью пакета модулей для работы с файлами «JPrime-files», состоящего из модулей, приведенных в Таблица 15

Таблица 15. Модули в составе модуля для работы с файлами

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-file-core | API информации о файлах |
| jprime-file-starter | Модуль для работы с хранилищем метаописания файлов |
| jprime-file-rest | Модуль с REST-методами работы с файлами |
| jprime-file-user-starter | Модуль метаописания файлов пользователя |
| jprime-file-user-rest | Модуль с REST-методами работы с файлами пользователя |
| jprime-file-user-scheduler | Модуль очистки старых файлов файлов пользователя по расписанию |
| jprime-filepreview-rest-starter | Модуль с общей логикой предоставления конвертации файлов по REST |
| jprime-filepreview-rest | Модуль скачивания файла для предпросмотра (для монолитной архитектуры) |
| jprime-filepreview-jpremote-rest | Модуль скачивания файла для предпросмотра (для микросервисной архитектуры) |
| jprime-file-pfdconverter-core | Базовый модуль конвертации файлов в PDF |
| jprime-file-pfdconverter-starter | Интерфейс конвертации файлов в PDF |
| jprime-file-pfdconverter-jpremote | Конвертации файлов в PDF: микросервисная (удаленная) реализация интерфейса PdfConverter из jprime-file-pfdconverter-starter |
| jprime-file-pfdconverter-local | Конвертации файлов в PDF: локальная реализация интерфейса PdfConverter из jprime-file-pfdconverter-starter |

## Модуль «jprime-file-core»

Модуль «jprime-file-core» предоставляет базовое API для передачи информации о файлах.

## Модуль «jprime-file-starter»

Модуль «jprime-file-starter» предназначен для работы с хранилищем метаописания файлов.

В модуле реализован сервис, который позволяет:

* получать информацию об указанном файле пользователя;
* производить отметки о скачивании файла;
* получать информацию обо всех файлах пользователя;
* удалять мету информацию о файле;
* сохранять новые данные о файлах;
* обновлять информацию о файле.

## Модуль «jprime-file-rest»

Модуль «jprime-file-rest» предоставляет REST API для работы с файлами. Для внешнего вызова доступны следующие REST-методы:

### Скачивание файла

GET /file/v1/<код файла>/<токен авторизации>

### Загрузка файла

POST /file/v1/upload

* body file=тело файла
* ответ

{

"fileCode" : "c7b12bec-ae16-4d4d-93f9-e49a0582c808"

}

## Модуль «jprime-file-user-starter»

Модуль «jprime-file-user-starter» извлекает метаописания файлов пользователя: информацию о файле, его расположении и т.д.

## Модуль «jprime-file-user-rest»

Модуль «jprime-file-user-rest» предоставляет REST API для работы с файлами пользователя. Для внешнего вызова доступны следующие REST-методы:

### Скачивание файла

GET /file/v1/<код файла>/<токен авторизации>

### Получение всех возможных файлов на скачивание

POST /file/v1/search

* запрос

{

"limit":"50",

"offset":"0"

}

* ответ

{

"offset": 0,

"limit": 50,

"totalCount": null,

"filesCount": 2,

"files": [

{

"fileCode": "17a0532a-c829-4900-9ee3-4773a3652cbc",

"name": "test1.txt",

"createdDate": "2018-11-28T13:58:25.764+0000",

"filledInDate": "2018-11-28T13:57:25.764+0000",

"length": 269

},

{

"fileCode": "31fc2257-8f66-46f0-8494-96717c7cd1de",

"name": "test2.txt",

"createdDate": "2018-11-28T13:58:20.140+0000",

"filledInDate": null,

"length": 269

}

]

}

### Загрузка файла

POST /file/v1/upload

* body file=тело файла
* ответ

{

"fileCode" : "c7b12bec-ae16-4d4d-93f9-e49a0582c808"

}

## Модуль «jprime-file-user-scheduler»

Модуль «jprime-file-user-scheduler» отвечает за очистку временных файлов пользователя, согласно cron-настройке.

## Модуль «jprime-filepreview-rest-starter»

Модуль «jprime-filepreview-rest-starter» содержит общую логику предоставления конвертации файлов по REST.

## Модуль «jprime-filepreview-rest»

Модуль «jprime-filepreview-rest» предоставляет REST API для загрузки файлов для предпросмотра в формате PDF/HTML (для монолитной архитектуры). Для внешнего вызова доступны следующие REST-методы:

### Cкачивание файла для предпросмотра

#### 1.1 Запрос атрибута типа файл

GET /filepreview/v1/<code>/<objectId>/file/<attrCode>/<bearer>

| Свойство | Описание |
| --- | --- |
| code | код метакласса |
| objectId | идентификатор объекта |
| attrCode | кодовое имя атрибута типа файл |
| bearer | токен авторизации |

Пример:

GET /filepreview/v1/eguApplicantDocs/614/file/fileName/<bearer>

#### 1.2 Запрос отдельного файла по fileCode

GET /filepreview/v1/<fileCode>/<bearer>

| Свойство | Описание |
| --- | --- |
| fileCode | GUID файла |
| bearer | токен авторизации |

Пример:

GET /filepreview/v1/647adcfc-ca35-4159-9d5e-62ef14a04461/<bearer>

#### 1.3 Запрос отдельного файла по storageCode и fileCode

GET /filepreview/v1/<storageCode>/<fileCode>/<bearer>

| Свойство | Описание |
| --- | --- |
| storageCode | Код хранилища |
| fileCode | GUID файла |
| bearer | токен авторизации |

#### Ответы

* 200 OK
* 415 UNSUPPORTED MEDIA TYPE - этот тип файла недопустим для предпросмотра, только прямое скачивание
* 422 UNPROCESSABLE ENTITY - сервер попытался конвертировать файл в PDF, но не смог (возможная причина: файл слишком большой и не успевает сконвертироваться в отведенное время)

### Получение допустимых форматов файлов для конвертации в PDF

GET /filepreview/v1/file-types

ответ:

{

"exts": [

"jpg",

"xlsx",

"tiff",

"rtf",

"csv",

"png",

"docx",

"tif",

"txt",

"pdf",

"doc",

"html",

"xls",

"odt",

"ods"

]

}

## Модуль «jprime-filepreview-jpremote-rest»

Модуль «jprime-filepreview-jpremote-rest» предоставляет REST API для загрузки файлов для предпросмотра в формате PDF (для микросервисной архитектуры). Для внешнего вызова доступны следующие REST-методы:

### Cкачивание файла для предпросмотра

#### 1.1 Запрос атрибута типа файл

GET /filepreview/v1/<code>/<objectId>/file/<attrCode>/<bearer>

| Свойство | Описание |
| --- | --- |
| code | код метакласса |
| objectId | идентификатор объекта |
| attrCode | кодовое имя атрибута типа файл |
| bearer | токен авторизации |

Пример:

GET /filepreview/v1/eguApplicantDocs/614/file/fileName/<bearer>

#### 1.2 Запрос отдельного файла по fileCode

GET /filepreview/v1/<fileCode>/<bearer>

| Свойство | Описание |
| --- | --- |
| fileCode | GUID файла |
| bearer | токен авторизации |

Пример:

GET /filepreview/v1/647adcfc-ca35-4159-9d5e-62ef14a04461/<bearer>

#### 1.3 Запрос отдельного файла по storageCode и fileCode

GET /filepreview/v1/<storageCode>/<fileCode>/<bearer>

| Свойство | Описание |
| --- | --- |
| storageCode | Код хранилища |
| fileCode | GUID файла |
| bearer | токен авторизации |

#### Ответы

* 200 OK
* 415 UNSUPPORTED MEDIA TYPE - этот тип файла недопустим для предпросмотра, только прямое скачивание
* 422 UNPROCESSABLE ENTITY - сервер попытался конвертировать файл в PDF, но не смог (возможная причина: файл слишком большой и не успевает сконвертироваться в отведенное время)
* остальные коды пробрасываются от сервиса, раздающего файлы, без изменений (напр. 403 FORBIDDEN)

### Получение допустимых форматов файлов для конвертации в PDF

GET /filepreview/v1/file-types

Ответ:

{

"exts": [

"jpg",

"xlsx",

"tiff",

"rtf",

"csv",

"png",

"docx",

"tif",

"txt",

"pdf",

"doc",

"html",

"xls",

"odt",

"ods"

]

}

## Модуль «jprime-file-pfdconverter-core»

Модуль «jprime-file-pfdconverter-core» содержит базовый API для конвертации файлов в PDF.

## Модуль «jprime-file-pfdconverter-starter»

Модуль «jprime-file-pfdconverter-starter» содержит базовый API для конвертации файлов в PDF.

## Модуль «jprime-file-pfdconverter-jpremote»

Модуль «jprime-file-pfdconverter-jpremote» представляет собой микросервисную реализацию интерфейса PdfConverter из «jprime-file-pfdconverter-starter» для конвертации файлов в PDF.

## Модуль «jprime-file-pfdconverter-local»

Модуль «jprime-file-pfdconverter-local» представляет собой локальную реализацию интерфейса PdfConverter из «jprime-file-pfdconverter-starter» для конвертации файлов в PDF.

# Пакет модулей работы с файловой системой «JPrime-fs»

Платформа JPrime имеет возможность взаимодействовать с файловой системой. Для этого используется модуль работы с файловой системой «JPrime-fs», набор модулей которого приведен в Таблица 16.

Таблица 16. Модули в составе модуля работы с файловой системой

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-dataaccess-fs | Модуль для ФС-хранилища |
| jprime-dataaccess-minio | Модуль для S3 хранилища minio |

## Модуль «jprime-dataaccess-fs»

Модуль «jprime-dataaccess-fs» реализует логику работы с ФС-хранилищем.

## Модуль «jprime-dataaccess-minio»

Модуль «jprime-dataaccess-minio» реализует логику работы с S3 хранилищем minio.

# Пакет модулей перенаправления запросов «JPrime-routes»

Модули пакета «JPrime-routes» умеют осуществлять гибкую маршрутизацию запросов при реализации микросервисной архитектуры в менеджер сервисов Eureka Server.

В модулях Платформы используется технология из стека Spring Cloud, согласно которой клиенты могут самостоятельно регистрироваться в службе обнаружения сервисов Eureka Server, а другие сервисы могут обращаться к Eureka Server для поиска необходимых им сервисов. Eureka Server хранит такую информацию как IP-адрес и порт сервиса.

Пакет состоит из модулей перенаправления запросов, представленных в Таблица 17:

Таблица 17. Модули в составе модуля перенаправления запросов

| Модуль | Описание модуля |
| --- | --- |
| jprime-route-adapter-core | Базовое API маршрутизации запросов |
| jprime-route-api-starter | Базовое API регистрации меты и утилит в Eureka Server |
| jprime-route-api-client | Регистрация обрабатываемой меты и утилит в Eureka Server |
| jprime-route-api-jpremoteclient | Регистрация проксируемой меты и утилит в Eureka Server |
| jprime-route-api-adapter | Route запросов и утилит согласно настроенной меты |
| jprime-route-file-starter | Базовое API регистрации файловых хранилищ в Eureka Server |
| jprime-route-file-client | Регистрация файловых хранилищ в Eureka Server |
| jprime-route-file-adapter | Route запросов согласно настроенным файловым хранилищам |
| jprime-route-meta-editor-starter | Базовое API регистрации конструкторов меты в Eureka Server |
| jprime-route-meta-editor-editor-client | Регистрация конструктора меты в Eureka Server |
| jprime-route-meta-editor-adapter | Route запросов конструкторов меты |
| jprime-route-reports-starter | Базовое API регистрации отчетов в Eureka Server |
| jprime-route-reports-client | Регистрация обрабатываемых отчетов в Eureka Server |
| jprime-route-reports-adapter | Route запросов согласно настроенным отчетам |
| jprime-route-reports-editor-starter | Базовое API регистрации конструкторов отчетов в Eureka Server |
| jprime-route-reports-editor-editor-client | Регистрация конструктора отчетов в Eureka Server |
| jprime-route-reports-editor-adapter | Route запросов конструкторов отчетов |

## Модуль «jprime-route-adapter-core»

Модуль «jprime-route-adapter-core» содержит базовое API маршрутизации запросов.

## Модуль «jprime-route-api-starter»

Модуль «jprime-route-api-starter» содержит базовое API регистрации метаописания и утилит в Eureka Server, содержит константы и общую логику для клиента и маршрутизации.

## Модуль «jprime-route-api-client»

Модуль «jprime-route-api-client» отвечает за регистрацию обрабатываемой меты и утилит в Eureka Server.

При регистрации меты модуль реализует логику передачи списка множественных кодовых имен меты, обрабатываемой сервисом.

При регистрации утилит модуль реализует логику передачи список кодовых имен утилит, обрабатываемой сервисом, а также список кодовых имен общих утилит, обрабатываемой сервисом.

## Модуль «jprime-route-api-jpremoteclient»

Модуль «jprime-route-api-jpremoteclient» отвечает за проксирования запросов на другие адреса, регистрацию проксируемой меты и утилит в Eureka Server.

## Модуль «jprime-route-api-adapter»

Модуль «jprime-route-api-adapter» осуществляет маршрутизацию запросов и утилит согласно настроенной меты и настроенным утилитам в Eureka Server.

## Модуль «jprime-route-file-starter»

Модуль «jprime-route-file-starter» содержит базовое API для регистрации файловых хранилищ в Eureka Server, содержит константы и общую логику для клиента и маршрутизации.

## Модуль «jprime-route-file-client»

Модуль «jprime-route-file-client» отвечает за регистрацию файловых хранилищ в Eureka Server.

При регистрации модуль реализует логику передачи кодового имени файлового хранилища.

## Модуль «jprime-route-file-adapter»

Модуль «jprime-route-file-adapter» осуществляет маршрутизацию запросов согласно настроенным файловым хранилищам в Eureka Server.

## Модуль «jprime-route-meta-editor-starter»

Модуль «jprime-route-meta-editor-starter» содержит базовое API для регистрации конструкторов меты в Eureka Server, содержит константы и общую логику для клиента и маршрутизации.

## Модуль «jprime-route-meta-editor-client»

Модуль «jprime-route-meta-editor-client» отвечает за регистрацию конструктора меты в Eureka Server

При регистрации модуль реализует логику передачи кодового имени сервиса-конструктора.

## Модуль «jprime-route-meta-editor-adapter»

Модуль «jprime-route-meta-editor-adapter» осуществляет маршрутизацию запросов конструкторов меты согласно настроенным конструкторам меты в Eureka Server.

## Модуль «jprime-route-reports-starter»

Модуль «jprime-route-reports-starter» содержит базовое API для регистрации отчетов в Eureka Server, содержит константы и общую логику для клиента и маршрутизации.

## Модуль «jprime-route-reports-client»

Модуль «jprime-route-reports-client» отвечает за регистрацию обрабатываемых отчетов в Eureka Server.

При регистрации модуль реализует логику передачи списка кодовых имен отчетов, обрабатываемой сервисом.

## Модуль «jprime-route-reports-adapter»

Модуль «jprime-route-reports-adapter» осуществляет маршрутизацию запросов согласно настроенному списку кодовых имен отчетов.

## Модуль «jprime-route-reports-editor-starter»

Модуль «jprime-route-reports-editor-starter» содержит базовое API для регистрации конструкторов отчетов в Eureka Server, содержит константы и общую логику для клиента и маршрутизации.

## Модуль «jprime-route-reports-editor-client»

Модуль «jprime-route-reports-editor-client» отвечает за регистрацию конструктора отчетов в Eureka Server.

## Модуль «jprime-route-reports-editor-adapter»

Модуль «jprime-route-reports-editor-adapter» осуществляет маршрутизацию запросов конструкторов отчетов в Eureka Server.

# Пакет модулей для работы с историей запросов «JPrime-requesthistory»

Платформа JPrime оснащена модулем для работы с историей запросов «JPrime-requesthistory», который состоит из модулей, представленных в Таблица 18:

Таблица 18. Модули в составе модуля для работы с историей запросов

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-requesthistory-core | Базовый модуль для работы с API |
| jprime-requesthistory-client-core | Модуль содержит сервис отправки истории запросов в различные брокеры сообщений |
| jprime-requesthistory-client-starter | Базовый модуль для клиентской работы с историей запросов |
| jprime-requesthistory-server-core | Модуль содержит сервис чтения истории запросов из Kafka |
| jprime-requesthistory-server-starter | Базовый модуль для серверной работы с историей запросов |
| jprime-requesthistory-server-jdbc | Модуль содержит сервис чтения истории запросов из базы данных |
| jprime-requesthistory-server-clickhouse | Модуль формирования запросов для ClickHouse |
| jprime-requesthistory-server-rest | Модуль для REST истории запросов |
| jprime-requesthistory-filter-core | Базовый модуль фильтров истории запросов |
| jprime-requesthistory-filter-jdbc | jdbc-модуль работы с фильтрами истории запросов |
| jprime-requesthistory-filter-rest | Модуль для REST фильтров истории запросов |

## Модуль «jprime-requesthistory-core»

Модуль «jprime-requesthistory-core» содержит основную логику по работе с историей запросов. Содержит все базовый API и реализации.

## Модуль «jprime-requesthistory-client-core»

Модуль «jprime-requesthistory-client-core» является базовым для клиентской работы с историей запросов, он содержит сервис отправки истории запросов в различные брокеры сообщений.

Для подключения клиента необходимо определиться с брокером сообщений (например, Kafka) и, в зависимости от выбранной реализации указать необходимые параметры.

## Модуль «jprime-requesthistory-client-starter»

Модуль «jprime-requesthistory-client-starter» подключается к «jprime-requesthistory-client-core» и содержит возможные реализации базового интерфейса RequestHistoryEventPublisher для публикации событий истории запросов.

Поддерживает следующие мосты для обмена сообщениями:

* Kafka.

## Модуль «jprime-requesthistory-server-core»

Модуль «jprime-requesthistory-server-core» содержит сервис чтения и обработки истории запросов, например из Apache Kafka. Полученные данные обрабатываются и загружаются в базу данных.

Поддерживаемые базы данных:

* ClickHouse.

## Модуль «jprime-requesthistory-server-starter»

Модуль «jprime-requesthistory-server-starter» для серверной работы с историей запросов, предоставляет реализацию RequestHistoryReader для чтения данных истории запросов. Полученные данные обрабатываются и загружаются в базу данных.

Поддерживаемые базы данных:

* ClickHouse.

Для использования нужно также подключить модуль, имеющий реализацию RequestHistoryReadRepository, например: «jprime-requesthistory-server-jdbc» (см. ниже)

## Модуль «jprime-requesthistory-server-jdbc»

Модуль «jprime-requesthistory-server-jdbc» содержит сервис чтения истории запросов из базы данных через jdbc-драйвер.

Поддерживаемые базы данных:

* ClickHouse.

При этом сам ClickHouse имеет встроенную возможность читать данные из Apache Kafka.

## Модуль «jprime-requesthistory-server-clickhouse »

Модуль «jprime-requesthistory-server-clickhouse» обеспечивает дополнительную обработку sql-запросов для ClickHouse. Обязателен к подключению для работы с данной базой данных.

## Модуль «jprime-requesthistory-server-rest»

Модуль «jprime-requesthistory-server-rest» предоставляет доступ к данным истории запросов через REST API: чтение и поиск.

Предоставляет доступ к данным по ключу и через поиск по отдельным модулям. Не предназначен для сложных запросов.

## Модули «jprime-requesthistory-filter-\*»

Модули группы «jprime-requesthistory-filter-\*» созданы для работы с фильтрами истории запросов.

Модуль «jprime-requesthistory-filter-core» содержит базовый интерфейс для фильтра истории запросов.

Модуль «jprime-requesthistory-filter-jdbc» является jdbc-реализацией логики работы с фильтрами истории запросов.

Модуль «jprime-requesthistory-filter-rest» предназначен для REST фильтров истории запросов.

# Пакет модулей работы с отчётностью «JPrime-reports»

В платформе JPrime в отдельный пакет модулей вынесены механизмы настройки и создания отчетов. Поддерживается создание как статистических отчетов, так и так и проинициализированной динамики. Пакет состоит из модулей, представленных в Таблица 19.

Таблица 19. Модули в составе модуля работы с отчётностью

|  |  |
| --- | --- |
| Код | Описание |
| jprime-reports-core | API настроек отчетов |
| jprime-reports-starter | Базовый модуль для настроек отчетов |
| jprime-reports-rest | Модуль для REST публикации отчетов |
| jprime-reports-creator-starter | Основной модуль создания отчетов |
| jprime-reports-creator-async | Модуль асинхронного создания отчетов с использованием metafilestorage |
| jprime-dynamicreports-core | Базовый модуль для динамических настроек отчетов |
| jprime-dynamicreports-core-jdbc | Базовый модуль для настроек динамических отчетов для работы с бд |
| jprime-dynamicreports-loader-starter | API для загрузчика настроек динамических отчетов |
| jprime-dynamicreports-loader-jdbc | Реализация загрузчика настроек динамических отчетов из БД |
| jprime-dynamicreports-editor-starter | API и REST-методы редактирования настроек динамических отчетов |
| jprime-dynamicreports-editor-imex | Утилиты экспорт/импорт настроек динамических отчетов |
| jprime-dynamicreports-editor-jdbc | Редактирования настроек и формирования динамических отчетов, хранящихся в БД |

## Модуль «jprime-reports-core»

Модуль «jprime-reports-core» содержит базовое API для передачи настроек отчетов.

## Модуль «jprime-reports-starter»

Модуль «jprime-reports-starter» является реализацией API, содеожащегося в модуле «jprime-reports-core».

## Модуль «jprime-reports-rest»

Модуль «jprime-reports-rest» предоставляет REST API для получения отчетов и их настроек. Для внешнего вызова доступны следующие REST-методы:

### Запрос на создание отчета

POST reports/v1/{report\_code}

* запрос

Параметры запроса всегда разные, они зависят от входных параметров, необходимых для создания отчета

Пример:

{

"ext": "xlsx"

}

* ответ

| Код | Описание |
| --- | --- |
| 400 | Неверный запрос |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Запрос отработал успешно |

{

"storageCode": "file",

"guid": "071397ea-79b0-4635-95fc-19f7dd0b4df7"

}

### Запрос на получение настроек отчетов

GET reports/v1/settings

* ответ

| Код | Описание |
| --- | --- |
| 400 | Неверный запрос |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Запрос отработал успешно |

[

{

"code": "testReport\_chartXlsx",

"description": "Test report chart xlsx",

"qName": "testReport.chartXlsx",

"type": "custom",

"jpClasses": [

"class1",

"class2"

],

"clientSearch": false,

"params": [

{

"code": "ext",

"type": "string",

"length": 10,

"description": "Расширение",

"qName": "report.ext",

"mandatory": true,

"multiple": false,

"value": "xlsx",

"refJpClass": null,

"refJpAttr": null,

"refFilter": null,

"clientSearch": true,

"enums": [

{

"value": "xlsx",

"description": "XLSX",

"qName": "report.ext.enum.xlsx"

},

{

"value": "pdf",

"description": "PDF",

"qName": "report.ext.enum.pdf"

}

]

},

{

"code": "param1",

"type": "integer",

"length": null,

"description": "Параметр1",

"qName": "report.param1",

"mandatory": false,

"multiple": false,

"value": null,

"refJpClass": null,

"refJpAttr": null,

"refFilter": null,

"clientSearch": false,

"enums": null

},

{

"code": "param2",

"type": "date",

"length": null,

"description": "Параметр2",

"qName": "report.param2",

"mandatory": false,

"multiple": false,

"value": null,

"refJpClass": null,

"refJpAttr": null,

"refFilter": null,

"clientSearch": false,

"enums": null

},

{

"code": "param3",

"type": "boolean",

"length": null,

"description": "Параметр3",

"qName": "report.param3",

"mandatory": false,

"multiple": false,

"value": false,

"refJpClass": null,

"refJpAttr": null,

"refFilter": null,

"clientSearch": false,

"enums": null

}

]

},

....

]

### Запрос на получение настроек определенного отчета

GET reports/v1/settings/{reportCode}

* ответ

| Код | Описание |
| --- | --- |
| 400 | Неверный запрос |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Запрос отработал успешно |

[

{

"description": "Заявление о предоставлении ГУ по организации профессиональной ориентации граждан",

"qName": "18",

"code": "20e6c388-10f3-4564-aeb2-2f9cb1efd6cb",

"clientSearch": true,

"params": [

{

"code": "reg\_num",

"type": "long",

"mandatory": true,

"multiple": false,

"value": null,

"refJpClass": "",

"refJpAttr": null,

"clientSearch": false,

"refFilter": {}

},

{

"code": "ext",

"type": "string",

"mandatory": true,

"multiple": false,

"value": "pdf",

"refJpClass": "",

"refJpAttr": null,

"clientSearch": false,

"refFilter": {}

}

]

}

]

### Получение описания всех отчетов

GET reports/v1/labels

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 500 | Ошибка сервера |
| 200 | Запрос отработал успешно |

[

{

"code": "creTest6",

"description": "creTest6",

"qName": "report.creTest6"

},

{

"code": "testReport\_chartXlsx",

"description": "Первый тестовый отчет",

"qName": "testReport.chartXlsx"

},

....

]

| Параметр | Описание |
| --- | --- |
| code | Код отчета |
| title | Название отчета |
| qName | QName названия отчета |

## Модуль «jprime-reports-creator-starter»

Модуль «jprime-reports-creator-starter» реализует основную логику создания статических и динамических отчетов.

Для формирования отчета необходимо наличие настройки, содержащей шаблон отчета, запросы для получения данных и описание параметров.

Описание настроек отчета может быть загружено через xml файл.

Формирование отчета осуществляется в прикладном коде.

В JPrime осуществлена поддержка двух генераторов отчетов:

* JasperReports;
* Yarg.

При указании шаблона отчета без расширения или с расширением .jrxml создание отчета будет производиться с помощью библиотеки JasperReports.

При наличии расширения, отличного от .jrxml и соответствующего общеупотребимым форматам ( doc,docx,xls,xlsx,pfd,csv,html) за создания отчетов будет отвечать шаблонизатор YARG.

## Модуль «jprime-reports-creator-async»

Задача модуля «jprime-reports-creator-async» получить запрос на создание, вызвать модуль «jprime-reports-creator-starter», сохранить итоговый файл отчета в нужное место, уведомить об окончании инициатора запроса.

## Модуль «jprime-dynamicreports-core»

Модуль «jprime-dynamicreports-core» является базовым для динамических настроек отчетов.

## Модуль «jprime-dynamicreports-core-jdbc»

Модуль «jprime-dynamicreports-core-jdbc» является реализаций динамических настроек отчетов в БД.

## Модуль «jprime-dynamicreports-loader-starter»

Модуль «jprime-dynamicreports-loader-starter» содержит API для загрузчика настроек отчетов.

## Модуль «jprime-dynamicreports-loader-jdbc»

Модуль «jprime-dynamicreports-loader-jdbc» является реализацией загрузчика настроек отчетов из БД.

## Модуль «jprime-dynamicreports-editor-starter»

Модуль «jprime-dynamicreports-editor-starter» содержит API и REST-методы редактирования настроек динамических отчетов, такие как:

### Получение списка конструкторов отчетов

GET /dynamicreports/v1/apps/

* ответ

{

"apps": [

{

"code": "reportseditor",

"title": "Конструктор отчетов"

}

]

}

| **Параметр** | **Описание** |
| --- | --- |
| **apps** | **Список конструкторов** |
| **code** | **Код конструктора** |
| **title** | **Название конструктора** |

### Получение списка группа отчетов

GET /dynamicreports/v1/reportGroups

* ответ

[

{

"code": "common",

"name": "Хранение в Minio S3",

"qName": "jpReportGroup.common"

}

]

| Свойство | Описание |
| --- | --- |
| code | Кодовое имя группы |
| name | Имя группы |
| qName | Полное имя группы |

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Логика отработала успешно |

### Получение списка хранилищ с данными

GET /dynamicreports/v1/reportStorages

* ответ

[

{

"code": "main",

"title": "Основное хранилище данных"

},

...

]

| Свойство | Описание |
| --- | --- |
| code | Кодовое имя хранилища |
| title | Заголовок хранилища |

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Логика отработала успешно |

### Инициализация настроек отчетов

POST dynamicreports/v1/dynamicreports-reportinit/mode/init

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 404 | Адрес не найден |
| 500 | Ошибка сервера |
| 200 | Логика отработала успешно |

{

"description": "Настройки отчетов проинициализированы",

"changeData": false,

"resultType": "message",

"qName": "dynamicreports-reportinit.init.ready"

}

### Создание настроек отчетов

POST /dynamicreports/v1/jpReports/

* headers

| header | Значение |
| --- | --- |
| Charset | [UTF-8](https://jira.metaprime.ru/browse/UTF-8) |

* запрос

| Свойство | Описание | Content Type |
| --- | --- | --- |
| reportSettings | Настройки отчета в json | application/json;charset=UTF-8 |
| templateData | Файл шаблона отчета |  |
| <произвольные имена> | Файл подшаблона отчета |  |

Если templateData = null, то шаблон отчета удаляется

### Обновление настроек отчетов

PUT /dynamicreports/v1/jpReports/

* headers

| header | Значение |
| --- | --- |
| Charset | [UTF-8](https://jira.metaprime.ru/browse/UTF-8) |

* запрос

| Свойство | Описание | Content Type |
| --- | --- | --- |
| reportSettings | Настройки отчета в json | application/json;charset=UTF-8 |
| templateData | Файл шаблона отчета |  |
| <произвольные имена> | Файл подшаблона отчета |  |

Если templateData = null, то шаблон отчета удаляется.

{

"guid": "c95ff2de-7106-48b6-b826-299e98172c77",

"code": "testReport",

...

"deletedSubtemplates": [

{

"guid": "edddb6d2-542d-410f-bbfa-b4533ed0bdae"

},

{

"guid": "abb65ca7-496c-493a-8bcf-6a3a077ab31f"

}

]

}

| Свойство | Описание |
| --- | --- |
| deletedSubtemplates | Информация об удаленных подшаблонах отчета |

### Удаление настроек отчетов

DELETE /dynamicreports/v1/jpReports/<гуид отчета>/

### Получение списка настроек отчетов

GET /dynamicreports/v1/jpReports/

### Получение определенных настроек отчетов

GET /dynamicreports/v1/jpReports/<гуид отчета>/

### Скачивание шаблона настроек отчетов

GET /dynamicreports/v1/jpReportTemplates/<гуид отчета>/<токен авторизации>

### Скачивание подшаблона настроек отчетов

GET /dynamicreports/v1/jpReportSubtemplates/<код подшаблона>/<токен авторизации>

### Поиск по отчётам

POST /dynamicreports/v1/jpReports/search

* запрос (пример с поиском по теме)

{

"filter": {

"or": [

{

"cond": {

"attr": "topicLink",

"exists": {

"cond": {

"attr": "topic",

"exists": {

"cond": {

"attr": "name",

"like": "{name}"

}

}

}

}

}

},

...

]

},

...

}

| Свойство | Описание |
| --- | --- |
| {name} | Наименование темы |

## Модуль «jprime-dynamicreports-editor-jdbc»

Модуль «jprime-dynamicreports-editor-jdbc» предназначен для редактирования настроек динамических отчетов, хранящихся в БД, через jdbc-драйвер.

## Модуль «jprime-dynamicreports-editor-imex»

Модуль «jprime-dynamicreports-editor-imex» содержит утилиты экспорта и импорта настроек динамических отчетов.

# Пакет модулей работы с утилитами «JPrime-utils»

В платформе JPrime реализована возможность работы с утилитами с помощью пакета модулей, представленных в Таблица 20.

Утилиты позволяют легко опубликовать любую бизнес-логику обработки данных в виде сервиса с описанием входных и выходных параметров.

Таблица 20. Модули в составе модуля для работы с утилитами

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-utils-core | API настроек утилит |
| jprime-utils-starter | Модуль, содержащий логику работы с утилитами |
| jprime-utils-rest | Модуль для REST публикации утилит |
| jprime-utils-uniutils | Модуль, содержащий типовые общие утилиты |

## Модуль «jprime-utils-core»

Модуль «jprime-utils-core» содержит базовое API для передачи настроек утилит.

## Модуль «jprime-utils-starter»

Модуль «jprime-utils-starter» содержит основную логику работы с утилитами.

## Модуль «jprime-utils-rest»

Модуль «jprime-utils-rest» предоставляет REST API для публикации утилит. Для внешнего вызова доступны следующие REST-методы:

### Универсальный вызов произвольного шага утилиты

POST /utils/v1/<код утилиты>/mode/<код шага утилиты>

#### Пример c REST запросом

Входные и выходные параметры произвольный REST, который сериализуется, десериализуется в параметры метода.

POST /utils/v1/hello/mode/response

* запрос

{

"request": "hello"

}

* ответ

{

"changeData": false,

"response": "Thanx for hello",

"resultType": "custom"

}

#### Пример c multipart/form-data запросом

POST /utils/v1/hello/mode/response

* запрос

"file": <binary>,

"request": "hello"

* ответ

{

"changeData": false,

"response": "Thanx for hello",

"resultType": "custom"

}

**Вызов общих (UNI) утилит**

При вызове общих утилит (uni = true) необходимо передать в теле запроса objectClassCode, по которому будет осуществляться роут на конкретный сервис, обрабатывающий этот класс меты.

Пример:

{

"objectClassCode": "eguApplicant",

.... //other required fields

}

**Режим check**

Если в утилите нет метода, помеченного code = "check", то по умолчанию код c mode/check отработает с 200 кодом и пустым результатом

{

"rootObjectClassCode": null,

"rootObjectId": null,

"objectClassCode": "universityNeat",

"objectIds": [2]

}

| Параметр | Описание |
| --- | --- |
| rootObjectClassCode | кодовое имя метакласса корневого объекта |
| rootObjectId | идентификатор корневого объекта |
| objectClassCode | Кодовое имя метаописания класса |
| objectIds | Идентификатор объекта |

### Получение описания всех утилит

* Доступ роль ADMIN

GET utils/v1/labels

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 500 | Ошибка сервера |
| 200 | Запрос отработал успешно |

[

{

"utilCode": "test2",

"modeCode": "ready",

"title": "Тестовая утилита (message ответ)",

"qName": ""

},

{

"utilCode": "test3",

"modeCode": "copy",

"title": "Тестовая утилита (объект ответ)",

"qName": ""

},

....

]

| Параметр | Описание |
| --- | --- |
| utilCode | Урл утилиты |
| modeCode | Код шага |
| title | Название шага |
| qName | QName шага |

### Получение списка всех утилит, доступных пользователю

GET utils/v1/settings

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 500 | Ошибка сервера |
| 200 | Запрос отработал успешно |

[

{

"utilCode": "perscopy",

"modeCode": "copy",

"title": "Копирование учетных данных ЛДПГУ",

"qName": "",

"confirmMessage": "Вы уверены ?",

"jpClasses": [

"pers"

],

"type": "object",

"jpAttrs": [

{

"jpClass": "pers",

"jpAttr": "payRef"

}

],

"inParams": [

{

"code": "rn",

"type": "long",

"description": "Регистрационный номер гражданина",

"qName": "perscopy.in.rn.title",

"mandatory": true,

"multiple": false,

"refJpClass": "pers",

"refJpAttr": "reg\_num",

"length": null

},

{

"code": "specCopy",

"type": "boolean",

"description": "Документы по образованию",

"qName": "perscopy.in.specCopy.title",

"mandatory": false,

"multiple": false,

"refJpClass": "",

"refJpAttr": "",

"length": null

},

...

],

"resultType": "custom",

"outCustomParams": [

{

"code": "rn",

"type": "long",

"description": "Регистрационный номер гражданина",

"qName": "perscopy.out.rn.title",

"mandatory": true,

"multiple": false,

"refJpClass": "pers",

"refJpAttr": "reg\_num",

"length": null

}

]

},

....

]

| Параметр | Описание |
| --- | --- |
| utilCode | Урл утилиты |
| modeCode | Код шага |
| title | Название шага |
| qName | QName шага |
| confirmMessage | Сообщение перед запуском шага |
| jpClasses | Список классов, для которых доступна утилита |
| type | Тип привязки утилиты (объект, список, произвольная) |
| jpAttrs | Настройки доступа на атрибутах |
| inParams | Список входных параметров |
| resultType | Тип исходящих параметров |
| outCustomParams | Список кастомных исходящих параметров |

### Получение списка всех утилит, доступных пользователю для указанного класса

GET utils/v1/settings/{classCode}

* ответ

| Код | Описание |
| --- | --- |
| 401 | Нет доступа |
| 500 | Ошибка сервера |
| 200 | Запрос отработал успешно |

[

{

"utilCode": "perscopy",

"modeCode": "copy",

"title": "Копирование учетных данных ЛДПГУ",

"qName": "",

"jpClasses": [

"pers"

],

"type": "object",

"inParams": [

{

"code": "rn",

"type": "long",

"description": "Регистрационный номер гражданина",

"qName": "perscopy.in.rn.title",

"mandatory": true,

"multiple": false,

"refJpClass": "pers",

"refJpAttr": "reg\_num",

"length": null

},

{

"code": "specCopy",

"type": "boolean",

"description": "Документы по образованию",

"qName": "perscopy.in.specCopy.title",

"mandatory": false,

"multiple": false,

"refJpClass": "",

"refJpAttr": "",

"length": null

},

...

],

"resultType": "custom",

"outCustomParams": [

{

"code": "rn",

"type": "long",

"description": "Регистрационный номер гражданина",

"qName": "perscopy.out.rn.title",

"mandatory": true,

"multiple": false,

"refJpClass": "pers",

"refJpAttr": "reg\_num",

"length": null

}

]

},

....

]

| Параметр | Описание |
| --- | --- |
| utilCode | Урл утилиты |
| modeCode | Код шага |
| title | Название шага |
| qName | QName шага |
| confirmMessage | Сообщение перед запуском шага |
| jpClasses | Список классов, для которых доступна утилита |
| type | Тип привязки утилиты (объект, список, произвольная) |
| jpAttrs | Настройки доступа на атрибутах |
| inParams | Список входных параметров |
| resultType | Тип исходящих параметров |
| outCustomParams | Список кастомных исходящих параметров |

### Массовая проверка доступности утилит

POST utils/v1/batchCheck

* запрос

{

"rootObjectClassCode": null,

"rootObjectId": null,

"ids": [

{

"objectClassCode": "universityNeat",

"objectIds": [2]

}

],

"utils": [

"test1", "test5", "test6", "test2", "test7"

]

}

| Параметр | Описание |
| --- | --- |
| rootObjectClassCode | кодовое имя метакласса корневого объекта |
| rootObjectId | идентификатор корневого объекта |
| ids | Идентификаторы проверяемых объектов |
| objectClassCode | Кодовое имя метаописания класса |
| objectIds | Идентификатор объекта |
| utils | Коды проверяемых утилит |

* ответ

{

"result": [

{

"objectClassCode": "universityNeat",

"objectId": "2",

"utilCode": "test1",

"result": {

"description": "Запуск разрешен",

"changeData": false,

"denied": false,

"resultType": "check",

"confirm": null,

"qName": null

}

},

{

"objectClassCode": "universityNeat",

"objectId": "2",

"utilCode": "test5",

"result": {

"description": "Запуск закрыт",

"changeData": false,

"denied": true,

"resultType": "check",

"confirm": null,

"qName": "utils.test5.denied"

}

}

],

"resultType": "batchCheck"

}

| Параметр | Описание |
| --- | --- |
| resultType | Тип ответа, всегда batchCheck |
| objectClassCode | Кодовое имя метаописания класса |
| objectIds | Идентификатор объекта |
| utilCode | Код проверяемой утилиты |
| result | Результат проверки |
| result.description | Сообщение проверки |
| result.qName | Код сообщения проверки |
| result.changeData | false |
| result.denied | Признак запрета запуска утилиты |
| result.resultType | Тип ответа, всегда check |
| result.confirm | Сообщение пользователю перед запуском утилиты |

## Модуль «jprime-utils-uniutils»

Модуль «jprime-utils-uniutils» содержит типовые общие утилиты, которые могут полезны независимо от прикладного решения, реализуемого на базе Платформы. Список таких утилит может дополняться со временем. Примером типовой общей утилиты является утилита массового изменения атрибутов, которая заменяет значения переданных атрибутов у объектов с определенным классом и идентификатором.

# Пакет модулей работы с адресами «JPrime-address»

Платформа JPrime обладает возможностями работы с адресами граждан как в формате ФИАС (Федеральной информационной адресной системы), так и в формате ГАР (Государственного адресного реестра). Данные функции реализованы в модулях работы с адресами, приведенных в Таблица 21.

Таблица 21. Модули в составе модуля работы с адресами

|  |  |
| --- | --- |
| Модуль | Описание модуля |
| jprime-fias-core | Базовый модуль работы с адресами |
| jprime-fias | Модуль работы с адресами ФИАС |
| jprime-gar | Модуль работы с адресами ГАР |

## Модуль «jprime-fias-core»

Модуль «jprime-fias-core» содержит базовую логику работы с адресами.

## Модуль «jprime-fias»

Модуль «jprime-fias» реализует логику работы с адресами формата ФИАС (Федеральная информационная адресная система).

## Модуль «jprime-gar»

Модуль «jprime-gar» реализует логику работы с адресами в формате ГАР (Государственного адресного реестра).