АКЦИОНЕРНОЕ ОБЩЕСТВО «СИТРОНИКС»

Руководство пользователя
ПО «Ситроникс. ГИС»

АННОТАЦИЯ

Настоящий документ представляет собой руководство программиста для программного обеспечения «Ситроникс.ГИС» (версия 1.32.1).

В документе приведены сведения о назначении и решаемых задачах Программы, условиях её применения, характеристиках, процедурах обращения к ней через API, а также описание входных и выходных данных.

Данный документ разработан в соответствии с требованиями следующих стандартов:

− ГОСТ 19.101-77 «Единая система программной документации (ЕСПД). Виды программ и программных документов»;

− ГОСТ 19.105-78 «ЕСПД. Общие требования к программным документам»;

− ГОСТ 19.504-79 «ЕСПД. Руководство программиста. Требования к содержанию и оформлению».

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ТЕРМИНОВ

В настоящем документе используются следующие сокращения и термины на русском и английском языках:

БД	–	База данных
ГИС	–	Геоинформационная система
ГОСТ	–	Государственный стандарт
ЕСИА	–	Единая система идентификации и аутентификации
ЕСПД	–	Единая система программной документации
ОЗУ	–	Оперативное запоминающее устройство (оперативная память)
ООП	–	Объектно-ориентированное программирование
ОС	–	Операционная система
ПКК	–	Публичная кадастровая карта
ПО	–	Программное обеспечение
САПР	–	Система автоматизированного проектирования
СУБД	–	Система управления базами данных
ЦП	–	Центральный процессор
API	–	Application programming interface, программный интерфейс приложения
ArcGIS	–	Семейство геоинформационных программных продуктов американской компании ESRI
CartoCSS	–	формальный язык для описания отображения геопространственных данных в картографических системах
CentOS	–	Дистрибутив Linux, основанный на коммерческом Red Hat Enterprise Linux компании RedHat и совместимый с ним
CSS	–	Cascading Style Sheets, формальный язык описания внешнего вида документа, написанного с использованием языка разметки
DELETE-запрос	–	Удаляет указанный ресурс

DPI	–	Dots Per Inch, параметр разрешающей способности (точек на дюйм)
EPSG	–	European Petroleum Survey Group, система классификации систем координат
GeoJSON	–	Открытый формат, предназначенный для хранения географических структур данных, основан на JSON
GeoTIFF	–	Открытый формат представления растровых данных в формате TIFF совместно с метаданными о географической привязке
GET-запрос	–	Используется для запроса содержимого указанного ресурса
GPS	–	Global Positioning System, система глобального позиционирования
HTML5	–	Язык для структурирования и представления содержимого всемирной паутины
HTTPS	–	HyperText Transfer Protocol Secure, расширение протокола HTTP для поддержки шифрования в целях повышения безопасности
JPG	–	Joint Photographic Experts Group, растровый графический формат
JS	–	JavaScript
JSON	–	JavaScript Object Notation, текстовый формат обмена данными, основанный на JavaScript
KML	–	Keyhole Markup Language, язык разметки на основе XML для представления трёхмерных геопространственных данных в программе «Google Планета Земля»
MIF	–	MapInfo Interchange Format, формат файла экспорта карты и базы данных программного продукта MapInfo
MongoDB	–	Документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц
MVT	–	Mapbox Vector Tiles, векторный формат для карт, описывающий геометрию, стилистику и т.д.

NGINX	–	Веб-сервер и почтовый прокси-сервер
OGC	–	Open Geospatial Consortium
OMJS	–	Ситроникс.ГИС Java Script
PATCH-запрос	–	Аналогично PUT, но применяется только к фрагменту ресурса
PNG	–	Portable Network Graphics, растровый формат хранения графической информации
POST-запрос	–	Применяется для передачи пользовательских данных заданному ресурсу
PostGIS	–	Открытое программное обеспечение, добавляющее поддержку географических объектов в реляционную базу данных PostgreSQL
PostgreSQL	–	Свободная объектно-реляционная система управления базами данных
PUT-запрос	–	Применяется для загрузки содержимого запроса на указанный в запросе URI
TMS	–	Tile Map Service
QGIS	–	Свободная кроссплатформенная геоинформационная система
REST	–	Representational State Transfer
RGB	–	Red, Green, Blue
RGBA	–	Red, Green, Blue, Alpha
SHP	–	Shapefile, векторный формат географических файлов
SPA	–	Single Page Application
SQL	–	Structured Query Language, декларативный язык программирования
TIFF	–	Tagged Image File Format, формат хранения растровых графических изображений
TinyOWS	–	WFS-сервер с поддержкой режима редактирования. Хранит данные в PostGIS

UTFGrid	–	спецификация растеризованных интерактивных данных
URL	–	Uniform Resource Locator, унифицированный указатель ресурса
uWSGI	–	веб-сервер и сервер веб-приложений для запуска приложений Python через протокол WSGI и его бинарный вариант uwsgi
WEB	–	Интернет-пространство
WebGL	–	Web-based Graphics Library, кроссплатформенный API для отображения 3D-графики в веб-браузере без использования плагинов
WFS	–	Web Feature Service, инструмент для получения или изменения описания функций
WMS	–	Web Map Service, стандартный протокол для обслуживания через Интернет географически привязанных изображений, генерируемых картографическим сервером на основе данных из БД ГИС
WMTS	–	Web Map Tile Service, стандартный протокол для обеспечения отображения изображения в мозаичном формате
WSGI	–	Web Server Gateway Interface – стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером

1 НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММ

1.1 Обозначение и наименование программы

Полное наименование программы: Программное обеспечение «Ситроникс.ГИС» (далее – Программа).

Условное обозначение программы: «Ситроникс.ГИС».

1.2 Назначение программы

Программа представляет собой современную геоинформационную платформу, предоставляющую большой набор инструментов по работе с пространственными данными – от сбора и хранения до публикации порталов федерального значения с отображением огромного количества визуализированной и структурированной информации.

1.3 Решаемые задачи программы

Программа обеспечивает решение следующих задач:

− автоматизированный сбор геопространственных данных из различных информационных источников с помощью разработанных ETL скриптов, в том числе – из автоматизированных систем сторонних производителей с поддерживаемыми API;

− работа с геопространственными данными из различных информационных источников, в том числе – из автоматизированных систем сторонних производителей с поддерживаемыми API;

− работа с тематическими геоинформационными сервисами;

− обработка и анализ векторных и растровых геопространственных данных, в том числе – по технологии векторной тайлизации;

− хранение (хостинг) исходных и обработанных геопространственных данных и аналитических отчётных материалов;

− поддержка распространённых наборов инструментов для проведения операций с геоданными: расчёты, публикация, интеграция, импорт и экспорт и др.;

− импорт и экспорт исходных и обработанных геопространственных данных и аналитических отчётных материалов;

− визуализация данных на карте с помощью встроенных инструментов;

− разработка, изменение и публикация картографической подложки с векторными и растровыми геопространственными данными;

− публикация геоданных и аналитических отчётных материалов;

− создание и управление учетными записями в Программе, а также настройки доступа к данным в соответствии с имеющимися правами;

− поддержка многоязычного интерфейса для пользователей;

− гибкое масштабирование, в зависимости от решаемых задач, технических возможностей инфраструктуры и наращивания функционала и объёма данных в будущем;

− создание и развитие веб-порталов с геопространственными данными (геопорталов) различного характера: географического, культурного, экономического, информационного, агротехнического, градостроительного, развлекательного и др.

1.4 Используемые технические средства

Операции обработки геопространственных данных требуют особых характеристик для используемого аппаратного обеспечения. Серверное ПО Программы допускается устанавливать и на физические, и на виртуальные серверные платформы.

Необходимое для работы Программы ПО указано в п. 1.2 настоящего документа.

Установка клиентской части Программы не требуется – веб-браузер автоматически выводит интерфейс и инструменты Программы при выборе её доменного адреса.

Установка и развёртывание серверного ПО Программы на аппаратном обеспечении выполняется двумя способами:

− все работы по установке, развёртыванию и настройке выполняют специалисты разработчика и производителя программных продуктов Ситроникс.ГИС – АО «Ситроникс» – в соответствии с имеющейся инфраструктурой и спецификациями серверов у Заказчика. В этом случае разработчиком предоставляется гарантийная техническая поддержка на Программу на один календарный год;

− специалисты Заказчика самостоятельно устанавливают, развёртывают и настраивают Программу и её составные части в соответствии с требованиями, предъявляемых к ПО.

Характеристики технических средств (серверов), необходимых для работы серверного ПО Программы, зависят от решаемых задач, нагрузки и количества пользователей. Минимальные и рекомендуемые характеристики, необходимые для работы серверного ПО Программы приведены в таблице 1.

При использовании одного сервера для работы всех составных частей Программы рекомендуется выбирать самую производительную конфигурацию (для обеспечения работы СУБД MongoDB).

Таблица 1 – Требования к характеристикам технических средств, необходимых для работы серверного ПО Программы

Сервер	**Минимальные характеристики**	**Рекомендуемые характеристики**
Серверное ПО	ЦП, не менее: 4 x 2.2 ГГц ОЗУ, ГБ, не менее: 16	ЦП, не менее: 12 x 2.9 ГГц ОЗУ, ГБ, не менее: 32

	Свободный объём HDD-диска, ГБ, не менее: 50 Сетевой интерфейс, Мбит/с: 100	Свободный объём HDD-диска, ГБ, не менее: 500 Сетевой интерфейс, Мбит/с: 1 000

_____

Примечание – ЦП: центральный процессор, ОЗУ: оперативное запоминающее устройство (оперативная память); указано рекомендуемое число физических ядер ЦП (не потоков)

Минимальные и рекомендуемые характеристики, необходимые для работы клиентского веб-приложения Программы приведены в таблице 2.

Таблица 2 – Требования к характеристикам технических средств, необходимых для работы клиентского веб-приложения Программы

Компьютер	**Характеристики минимальные**	**Характеристики рекомендуемые**
Клиентское ПО	ЦП: 2-ядерный, частотой не менее 3 ГГц ОЗУ, ГБ, не менее: 4 Свободный объём HDD-диска, ГБ, не менее: 10 Сетевой интерфейс, Мбит/с: 100	ЦП: 4-ядерный, частотой не менее 3,3 ГГц ОЗУ, ГБ, не менее: 8 Свободный объём HDD-диска, ГБ, не менее: 40 Сетевой интерфейс, Мбит/с: 1 000

_____

Примечание – ЦП: центральный процессор, ОЗУ: оперативное запоминающее устройство (оперативная память)

1.5 Используемые программные средства

1.5.1 Серверное программное обеспечение

Стабильная работа Программы требует наличия следующего серверного программного обеспечения (далее – ПО):

− операционная система (далее – ОС): Ubuntu, Debian, CentOS, Red Hat Enterprise Linux;

− системное ПО:

1) Docker – система управления программными контейнерами. Обеспечивает автоматизацию процессов развёртывания и управления модулей Программы (версия не ниже 19.03.5);

2) Docker Compose – компонент ПО Docker (версия не ниже 1.24.0);
3) Nginx – веб-сервер и почтовый прокси-сервер для распределения сетевых запросов между модулями Программы (версия не ниже 1.18);
4) uWSGI – веб-сервер и сервер веб-приложений.

- системы управления базами данных (далее – СУБД):
  1) PostgreSQL (версия не ниже 12.1);
  2) PostGIS – расширение СУБД PostgreSQL для работы с геоданными (версия не ниже 2.6/3.0);
  3) MongoDB – документоориентированная СУБД для хранения кэшированных данных (версия не ниже 4.0.3, но не выше платной).
- сервисы: Go App – сервис кэширования тайловых карт;
- фреймворки:
  1) Flask – фреймворк веб-приложений (версия не ниже 1.1.1);
  2) React – фреймворк HTML5 веб-приложений (версия не ниже 16.8.0).
- программные библиотеки:
  1) Mapnik – библиотека для рендеринга (отрисовки) растровых карт (версия не ниже 3.0.21);
  2) GDAL/OGR – библиотека для чтения и записи растровых и векторных геопространственных форматов данных (версия не ниже 2.4.0);
  3) OpenLayers – библиотека, предназначенная для создания карт на основе программного интерфейса (версия не ниже 6.1.1).
- интерпретаторы языков программирования:
  1) Python (версия не ниже 3.6.9);
  2) Go (версия не ниже 1.12).

Вышеуказанное ПО может быть развернуто на одном или нескольких физических (или виртуальных) серверах.

Примечание – Для увеличения производительности рекомендуется использовать несколько серверов с распределенной нагрузкой между ними.

1.5.2 Клиентское программное обеспечение

Стабильная работа Программы требует наличия следующего клиентского ПО:

- операционная система:
    1) на платформе ядра Linux – Ubuntu, Debian, CentOS, Fedora, Alt Linux, Oracle Linux, Red Hat, SUSE;
    2) на платформе ядра компании Microsoft – Windows 7 и выше;
    3) на платформе ядра компании Apple – Mac OS.
- веб-браузер для компьютера или ноутбука:
    1) Microsoft Edge (версии не ниже 80);
    2) Mozilla Firefox (версии не ниже 73);
    3) Google Chrome (версии не ниже 80);
    4) Apple Safari (версии не ниже 14);
    5) Opera (версии не ниже 67);
    6) Яндекс.Браузер (версии не ниже 20).
- веб-браузер для мобильных устройств (планшет, смартфон):
    1) Apple iOS Safari (версии не ниже 14);
    2) Android Google Chrome (версии не ниже 80).

Примечание – Работа с веб-браузерами для мобильных устройств поддерживается только публичной частью клиентского веб-приложения.

2 ХАРАКТЕРИСТИКА ПРОГРАММЫ

2.1 Описание структуры

Структура Программы реализована в модульном исполнении, при котором модули Программы взаимодействуют между собой посредством прикладных интерфейсов программирования (API). Для взаимодействия со сторонними источниками геопространственной и картографической информации используются протоколы стандарта OGC (WMS, WFS, WMTS). Модульная реализация повышает надёжность и отказоустойчивость Программы, а также упрощает развитие и добавление новой функциональности при расширении объёма решаемых задач.

Структура серверного ПО

Структура серверного ПО Программы показана на рисунке 1. Все модули работают на технологиях контейнеров ПО Docker. Под наименованиями модулей на рисунке 1 указаны используемые ими в работе программные библиотеки и приложения. Описание используемого ПО Программы приведено в п. 1.5 настоящего документа.

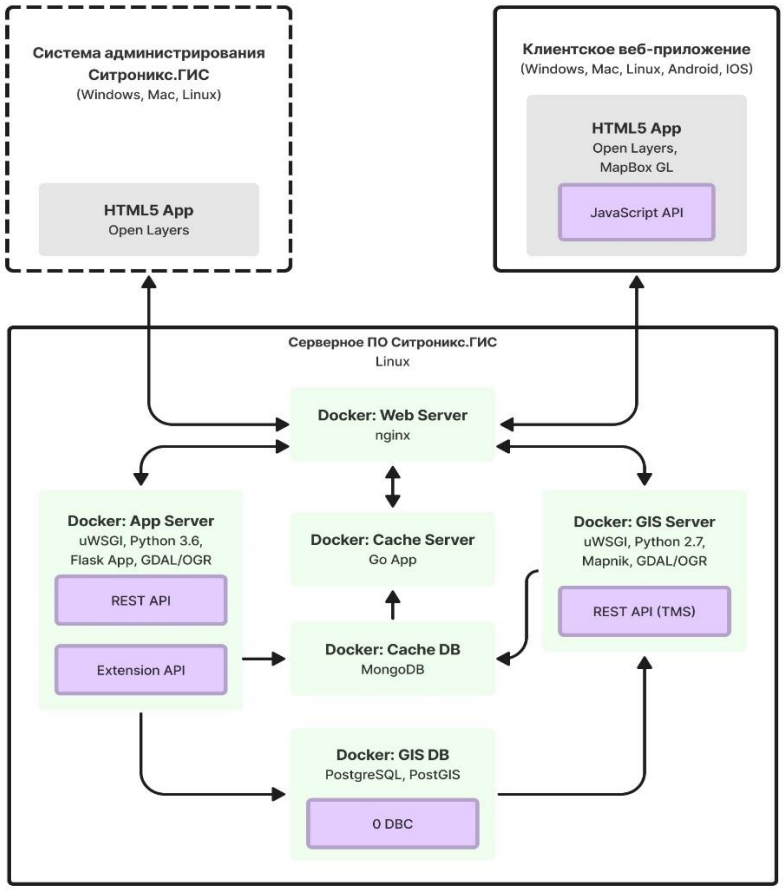

Рисунок 1 – Структура серверного ПО Программы

Структура серверного ПО включает в себя следующие модули:

− модуль управляющего веб-сервера «Web Server» на базе ПО NGINX;

− модуль приложений «App Server» с поддержкой интерфейсов взаимодействия REST API и Extensions API;

− модуль кэширования геоданных «Cache Server»;

− модуль отрисовки картографии «GIS Server» с поддержкой интерфейса взаимодействия REST API и технологией Tile Map Service (TMS);

− «Cache DB» – модуль взаимодействия с СУБД MongoDB (работа с кэшированными геоданными);

− «GIS DB» – модуль взаимодействия с СУБД PostgreSQL (работа с некэшированными геоданными);

− клиентское веб-приложение (на рис. 1 показано областью «Клиентское веб-приложение»).

Управление параметрами выполняется через клиентское ПО Программы (на рис. 1 показано областью со штриховыми границами).

Входящие в структуру системы и сервисы описаны ниже.

Структура клиентского ПО

Структура клиентского ПО Программы показана на рисунке 2. Описание используемого ПО Программы приведено в п. 1.5 настоящего документа.

Структура клиентского ПО включает в себя следующие модули:

- модуль управления геоданными;
- модуль настройки публикации;
- модуль отображения геоданных;
- модуль управления доступом.

Областью, обозначенной штриховыми границами, указано серверное ПО Программы (см. рисунок 2).

Пользователь получает доступ к Программе через веб-браузер. Веб-браузер отображает графический интерфейс Программы и предоставляет доступ к инструментам для работы. Общий вид интерфейса Программы показан на рисунке 3.

Входящие в Программу модули и сервисы описаны ниже.

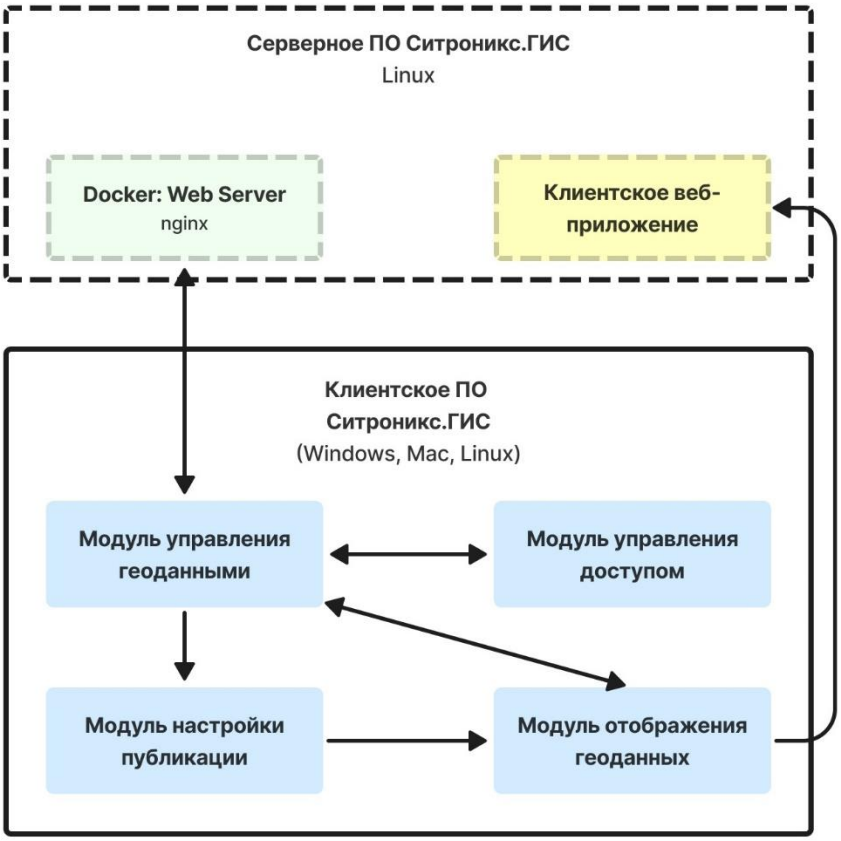

Рисунок 2 – Структура клиентского ПО Программы

Рисунок 3 – Интерфейс Программы

Описание работы с клиентской частью Программы приведено в документе «Ситроникс.ГИС». Руководство оператора».

2.1.1 Система кэширования

Система кэширования написана на языке программирования Go и предназначена для повышения производительности работы с геопространственными данными.

Система кэширования в режиме реального времени анализирует запросы для рендеринга (отрисовки) геоданных (в виде тайлов) и помещает отрисованные запрашиваемые данные в СУБД MongoDB для хранения. СУБД MongoDB является документоориентированной базой данных и обеспечивает высокую скорость обработки запросов.

Использование системы кэширования позволяет выдавать уже подготовленные данные пользователю, избавляя от необходимости повторного процесса поиска, загрузки и рендеринга, что значительно сокращает время предоставления данных и снижает нагрузку на серверную инфраструктуру.

В системе кэширования используются два уровня кэширования данных – так называемые «холодный» и «горячий» кэши. «Холодный» кэш организован в структуре хранилища СУБД MongoDB и геоданные хранятся там постоянно. «Горячий» кэш сформирован в оперативной памяти (ОЗУ) сервера, на котором установлена система кэширования. Скорость доступа к оперативной памяти намного выше, чем скорость доступа к хранилищам на магнитных и твердотельных носителях, поэтому выдача отрисованных геоданных из оперативной памяти происходит в кратчайшие сроки. Запрашиваемые из «холодного» кэша геоданные помещаются в «горячий» кэш и хранятся определённый период времени, после чего удаляются до нового запроса.

Система кэширования обладает механизмом распределения параллельных потоков выдачи данных, который выравнивает производительность системы в моменты пиковых нагрузок и предотвращает формирование узких мест в потоках (деградацию производительности).

Система кэширования содержит механизм инвалидации кэша – механизм, позволяющий отслеживать актуальность кэшированных данных, благодаря чему неактуальный кэш удаляется и, при повторном обращении пользователя, позволяет инициировать отрисовку (рендер) и обновление

только тех участков, в которых произошли изменения (т. н. точечная инвалидация).

Для увеличения производительности система кэширования содержит общий (единый) кэш для всех пользователей с разделением по уровню доступа к данным.

Система кэширования содержит механизм предварительной подготовки (т. н. «прогрева») кэша для указанных географических территорий и масштабов (зумов). Администратор Программы может предварительно сформировать отрисованные тайлы для наиболее часто используемых пользователями областей карты (область, город, район и т. д.) и их масштабы. Предварительно кэшированные («прогретые») отрисованные тайлы обеспечивают высокую скорость выдачи геоданных пользователю.

2.1.2 Система серверного рендеринга

Система серверного рендеринга предназначена для подготовки и выдачи уже сформированных данных (или основную их часть) в клиентское приложение для отображения в веб-браузере пользователя.

Для реализации процесса рендеринга растровых тайлов, векторных тайлов и тайлов UTFGrid разработаны веб-приложения на языках программирования Python и Go.

Система серверного рендеринга является многопоточной и может быть развёрнута на одном элементе вычислительной инфраструктуры (виртуальном или физическом сервере) или на кластере вычислительных ресурсов.

При работе системы серверного рендеринга на нескольких серверах один из серверов назначается главным (управляющим) и на него устанавливается управляющий сервис – балансировщик задач («worker»). Остальные серверы являются исполнительными и выполняют непосредственно рендеринг данных. Управляющий сервер также может выполнять рендеринг – одновременно с управлением он будет работать с геоданными как исполнительный сервер, так как сервис балансировщика не требует значительных вычислительных ресурсов.

Балансировщик обеспечивает приём множества входящих запросов (задач на рендеринг), их перераспределение на исполнительные серверы и контроль выполнения работы. Сформированные исполнительными серверами данные после рендеринга перенаправляются к пользователю также через балансировщик.

Балансировка рендеринга данных выполняется тонкой настройкой параметров конфигурационного файла в зависимости от количества процессоров и ядер в исполнительных серверах. На каждом исполнительном сервере определяется количество выполняющих рендеринг ядер и балансировщик контролирует их загруженность и выполнение задач. При избыточном входящем количестве задач балансировщик формирует очередь и перенаправляет очередные задачи на исполнительные сервера при освобождении их процессоров или ядер.

Система серверного рендеринга (отрисовки) геопространственных данных использует двухуровневую схему организации рендера. При отсутствии тайлов в кэше запрос на отрисовку поступает в специальный управляющий компонент (балансировщик), в котором происходит обработка запроса и проверка необходимых разрешений (права доступа), после чего выполняется отправка запроса в сервис отрисовки (рендера).

Система серверного рендеринга имеет возможность гибкой настройки параметров рендера (отрисовки) данных:

- поддержка различных форматов кодирования выходных растровых тайлов (PNG, PNG8);
- условия использования метатайлов (применимость и размер);
- условия упрощения геометрии (методами «simplify»);
- условия генерации векторных тайлов в бинарном формате MVT (MapBox Vector Tiles).

В системе серверного рендеринга реализован механизм отрисовки тайлов для различных форматов дисплеев. Для больших дисплеев и дисплеев с высокой плотностью точек (DPI) системой генерируются так называемые retina-тайлы. Поддерживается несколько разновидностей дисплеев высокой плотности – x2, x3 и другие.

2.1.3 Модуль управления геоданными

Модуль управления геоданными обеспечивает реализацию функций Программы, позволяющих работать с геоданными карты (слоя) – подготовка, создание, изменение, удаление. Для реализации этого функционала модуль взаимодействует с серверным ПО программы.

2.1.4 Модуль настройки публикации

Модуль настройки публикации обеспечивает реализацию функций Программы по настройке параметров публикации, а также позволяет добавить, настроить или удалить дополнительные виджеты, расширяющие функционал публичной карты.

2.1.5 Модуль отображения геоданных

Модуль отображения геоданных обеспечивает визуальное представление пространственных данных и взаимодействия с ними как в клиентском веб-приложении, так и в интерфейсе Программы.

Описание процессов отображения геоданных и работы с ними в клиентском веб-приложении приведено в документе «Программное обеспечение «Ситроникс.ГИС». Руководство оператора».

2.1.6 Модуль управления доступом

Модуль управления доступом обеспечивает реализацию функций Программы при создании, изменении и удалении субъектов доступа (группы и пользователи с разными уровнями прав) и объектов доступа (каталоги, карты, слои, столбцы таблицы (колонки), объекты слоя).

3 ОБРАЩЕНИЕ К ПРОГРАММЕ

3.1 Настройка работы программы через REST API

3.1.1 Описание параметров

Функции API, участвующих в URL запросе, содержат следующие параметры:

- «project_code» – код каталога;
- «map_code» – код карты (совпадает с названием схемы PostgreSQL);
- «layer_code» – код слоя (совпадает с названием таблицы PostgreSQL);
- «field_id» – идентификатор (ID) колонки структуры слоя;
- «object_id» – идентификатор объекта (совпадает со значением колонки «gis_id» в таблице PostgreSQL);
- «file_id» – идентификатор файла;
- «file_uuid» – uuid файла.

Сервер принимает GET, POST, PUT, PATCH, DELETE запросы. Обратиться можно только к карте, для которой включена публикация с учётом прав доступа и активной сессии. Все функции редактирования данных в обязательном порядке требуют предварительной авторизации.

Код каталога (project) выводится в верхней панели в строке навигации первым элементом. Если указан код несуществующего каталога, то сервер вернет ошибку со статусом 500. Описание ошибок приведено в п. 3.1.12.

Дополнительные необязательные параметры всех функций API содержат:

- f = json, pdf, fjson: формат ответа (по умолчанию json); формат pdf доступен исключительно для функции получения информации по объекту; формат fjson позволяет форматировать выводимые данные согласно настройкам форматирования;
- lng = <код языка>: указывает код языка для запроса. Если не указан – запрашивается основной язык карты. Язык учитывается в названиях

контейнеров и колонок. При этом коды колонок и служебные константы имени сущностей (name) будут приходить без суффиксов кода языка и без них же должны отсылаться на сервер.

3.1.2 Аутентификация

3.1.2.1 Аутентификация на проекте

Запрос:

POST http://<host>/api/2.7/<project_code>/login/

Здесь и далее по тексту принято, что за авторизованную сессию отвечает только параметр «token». Он выдаётся как результат функции аутентификации и является идентификатором сессии. Сессия хранится на сервере в течение десяти минут после последнего запроса с её участием, после чего автоматически уничтожается.

Идентификатор сессии необходим при выполнении любых операций создания/изменения/удаления с использованием REST API. Для этого идентификатор сессии нужно передать в параметре «token» запроса. Результат выполнения любых операций REST API (как получения, так и изменения) зависит от настроек прав доступа пользователей в Программе. Если «token» не передан в функцию, результат будет зависеть от настроек прав доступа публичного пользователя, в другом случае – от настроек прав доступа пользователя, определяемого переданным «token».

Обязательные параметры запроса:

- user – логин пользователя;
- pass – пароль пользователя.

Ответ:

```
{
   "token": "Токен авторизации"
}
```

Если указан несуществующий логин или неверный пароль для существующего логина, то сервер вернет ошибку со статусом 403. Описание ошибок приведено в п. 3.1.12.

3.1.2.2 Проверка авторизации

Запрос:

GET http://<host>/api/2.7/<project_code>/authorized/

Обязательные параметры запроса: token – идентификатор сессии.

Ответ: True.

Если пользователь авторизован в системе администрирования (но не через API), или не авторизован – вернет ответ False.

3.1.2.3 Завершение авторизационной сессии

Запрос:

POST http://<host>/api/2.7/<project_code>/logout/

Обязательные параметры запроса: token – идентификатор сессии.

Ответ

```
{
   "token": ""
}
```

3.1.3 Работа с картами

Карта, созданная в редакторе карт Программы, доступна для работы только в административном интерфейсе Программы. Для предоставления к такой карте доступа внешним пользователям её необходимо опубликовать. Публикация карты в разделе «Публичная карта» выполняется в клиентском ПО Программы.

Без публикации, созданной в редакторе карты, доступ к ней через REST API невозможен, при этом созданная посредством API карта автоматически публикуется.

3.1.3.1 Создание карты

Для выполнения запроса на создания необходим токен авторизации (см. выше).

Запрос:

POST http://<host>/api/2.7/<project_code>/

Параметры создаваемой карты передаются в формате json в теле запроса:

```
{
   «code»: "new_map",
   «name»: "Новая карта",
   «css_style»: "CartoCSS-стиль отображения карты – подложка для рисования слоев"
}
```

где:

− «code» (**обязательный**) – код создаваемой карты;

− «name» (*необязательный*) – имя карты. Если не указано, то не будет заполнено;

− «css_style» (*необязательный*) – CartoCSS-стиль отображения карты. Если не указан, то будет использован стиль отображения по умолчанию.

Ответ:

```
{
   "id": 1,
   "code": "map1",
   "name": "Карта 1",
```

"css_style": "CartoCSS-стиль отображения карты – подложка для рисования слоев",

"group_layers": 0,

"has_publication": true

}

где:

− id – идентификатор карты;

− code – код карты;

− name – имя карты. Если не указано, то не будет заполнено;

− css_style – CartoCSS-стиль отображения карты;

− group_layers – тип отображения карты (0 – несколькими слоями, 1 – единым слоем);

− has_publication – признак публикации карты. Если значение равно true – карта доступна для просмотра без авторизации.

Если указан код несуществующей карты, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.3.2 Получение объекта карты

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/

Здесь и далее по тексту принято: если карта не опубликована, и пользователь не авторизовался – система вернет ошибку доступа со статусом 403. Описание ошибок приведено в п. 3.1.12.

Ответ:

{

"id": 1,

"code": "map1",

"name": "Карта 1",

"css_style": "CartoCSS-стиль отображения карты – подложка для рисования слоев",

"group_layers": 0,

"has_publication": true

}

где:

- id – идентификатор карты;
- code – код карты;
- name – имя карты. Если не указано, то не будет заполнено;
- css_style – CartoCSS-стиль отображения карты;
- group_layers – тип отображения карты (0 – несколькими слоями, 1 – единым слоем);
- has_publication – признак публикации карты. Если значение равно true – карта доступна для просмотра без авторизации.

Если указан код несуществующей карты, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.3.3 Получение дерева слоёв

Дерево слоёв используется виджетом «Список слоёв», обеспечивающий настройку включаемых слоёв карты, редактирование их структуры и назначение им иконок. Виджет является частью клиентского ПО Программы.

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/

Необязательные параметры запроса:

- type:
    1) full – получить полное дерево слоёв с иерархией и порядком для отрисовки;
    2) publish (по умолчанию) – получить "дерево публикации".

− returnCss:

1) 1 – включить в ответ CSS стили слоя;
2) 0 (по умолчанию) – не включать в ответ CSS стили слоя.

Ответ:


```
[
  {
    "id": 1,
    "type": "layer",
    "code": "layer1",
    "name": "Слой 1",
    "sort": 1,
    "parent_id": 0,
    "selected": false,
    "map_layer_id": 1,
    "icon": "static/path/icon.png",
    "render_type": 0
  },
  ...
]
```


При запросе "дерева публикации" (значение параметра type = publish) в ответ попадут только опубликованные слои. При запросе полного дерева (значение параметра type = full) в ответ попадут все слои текущей карты.

Если указан код несуществующей карты, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

Ответ:

Структура каждого элемента в ответе на запрос дерева слоёв соответствует ответу на запрос «Получение слоя карты» (см. п. 3.1.4.2), кроме элементов с типом «folder», которые являются папками.

3.1.3.4 Сохранение дерева публикации

Дерево слоёв публикации – это наполнение виджета «Список слоёв». Для выполнения запроса необходим токен авторизации.

Запрос:

POST http://<host>/api/2.7/<project_code>/<map_code>/publish/

Формат запроса:


```
[
  {
    "id": 1,
    "sort": 1,
    "parent_id": 0,
    "selected": true,
    "map_layer_id": 1,
    "icon": "static/path/icon.png"
  },
  {
    "id": 2,
    "sort": 1,
    "parent_id": 1,
    "selected": true,
    "map_layer_id": 2,
    "icon": "static/path/icon.png"
  },
  {
    "id": 3,
    "sort": 2,
    "parent_id": 1,
    "selected": true,
    "map_layer_id": 3,
    "icon": "static/path/icon.png"
```

```
  },
  ...
]
```

где:

− id (обязательный) – уникальный идентификатор элемента дерева публикации. Используется в свойстве "parent_id" дочерних элементов для создания древовидности;

− sort (необязательный) – сортировка слоя в дереве публикации;

− parent_id (обязательный) – идентификатор ("id") родительского элемента. 0 указывает на корень дерева;

− selected (необязательный) – флаг, включен ли элемент по умолчанию в дереве публикации, значение по умолчанию = true;

− map_layer_id (обязательный) – идентификатор слоя в системе (векторный/растровый/виртуальный), на основе которого создается элемент дерева;

− icon (необязательный) – иконка для отображения элемента на карте.

Примечание: Параметр «id» должен быть целочисленным и уникальным в пределах одного запроса.

Ответ:

Формат ответа совпадает с функцией «Получение дерева слоёв» (см. выше) с параметрами по умолчанию.

3.1.3.5 Получение начальных настроек публичной карты

Запрос:

GET http://<host>/api/2.7/<project_code>/

Формат запроса:

```
{
  "map_code": "new_map",
}
```

Полученный в ответе файл конфигурации инициализирует первоначальную работу набора Ситроникс.ГИС JS API.

3.1.4 Работа со слоями

Слой карты содержит объекты с геопространственными данными и должен принадлежать карте. Слои бывают нескольких типов:

- векторный – значение поля "type" = layer;
- виртуальный – значение поля "type" = virtual;
- материализованный – значение поля "type" = material;
- растровый – значение поля "type" = raster.

По типу геопространственных данных слои карты подразделяются на следующие:

- Point – содержит геопространственные данные, представляющие собой точки на карте. Значение поля "g_type" = Point;
- LineString – содержит геопространственные данные, представляющие собой линии на карте. Значение поля "g_type" = LineString;
- Polygon – содержит геопространственные данные, представляющие собой полигоны на карте. Значение поля "g_type" = Polygon;
- Unknown – содержит геопространственные данные, представляющие собой любой из вышеперечисленных типов геопространственных данных на карте. Значение поля "g_type" = Unknown.

3.1.4.1 Создание слоя карты

Для выполнения запроса необходим токен авторизации.

Запрос:

POST http://<host>/api/2.7/<project_code>/<map_code>/layers/

Параметры создаваемого слоя в формате json должны передаваться в теле запроса:

{

"code": "new_layer",
"localized_name": true,
"name": {"ru": "Новый слой", "en": "New Layer"},
"parent_id": 2,
"sort": 3.5,
"css_style": "CartoCSS-стиль отображения слоя"
}

где:

− code (необязательный) – код создаваемого слоя. Если не указан, то будет сгенерирован автоматически;

− localized_name (необязательный) – флаг, локализовать ли название слоя. Значение по умолчанию = false;

− name (необязательный) – имя слоя или массив локализованных названий слоя в формате {"ru": "Новый слой", "en": "New Layer"}, где ru и en – коды языков локализации (в случае если выставлен флаг localized_name);

− parent_id (необязательный) – идентификатор родительского контейнера (карта/папка), в котором будет находиться слой в дереве. По умолчанию слой будет создаваться непосредственно в указанной (в url-пути) карте;

− sort (необязательный) – значение сортировки для создаваемого слоя. Если параметр не указан, то по умолчанию слой будет добавляться последним для указанного parent_id. При изменении сортировки сервер выполнит пересортировку всех элементов в пределах указанного parent_id, в результате чего значение сортировки для каждого контейнера будет целочисленным;

− css_style (необязательный) – CartoCSS-стиль отображения слоя. В случае отсутствия значения будет использован стиль отображения по умолчанию.

Ответ:

Структура ответа на запрос создания слоя соответствует ответу на запрос «Получение слоя карты».

Если указан несуществующий идентификатор родительского контейнера, или родительский контейнер принадлежит другой карте, то сервер вернет ошибку со статусом 404.

Если указан не валидный CartoCSS-стиль отображения слоя, то сервер вернет ошибку со статусом 500.

Описание ошибок приведено в п. 3.1.12.

3.1.4.2 Получение слоя карты

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>

Ответ:

{
"id": 1,
"type": "layer",
"code": "layer1",
"name": "Слой 1",
"sort": 1,
"parent_id": 0,
"selected": false,
"map_layer_id": 1,
"icon": "static/path/icon.png",
"g_type": "Point|LineString|Polygon|Unknown",
"css_style": "CartoCSS-стиль отображения слоя",
"render_type": 0,
"style_mode": 1
}

где:

− id – уникальный идентификатор слоя. Используется в свойстве "parent_id" дочерних элементов для создания древовидности;

− type – тип слоя. Поскольку запрашивался слой карты – значение установлено "layer";

− code – код слоя. Уникален для всех слоёв в пределах карты;

− name – имя слоя. Может не иметь значения. Если значение не заполнено, то отображаться будет код карты;

− sort – значение сортировки слоя в дереве;

− parent_id – идентификатор родительского контейнера (карта/папка), в котором находится слой в дереве;

− selected – признак, включен ли слой для отображения содержащихся в нем геопространственных данных на карте;

− map_layer_id – уникальный идентификатор слоя – имеет то же значение, что и свойство id и используется для совместимости с более старыми версиями;

− icon – иконка для отображения элемента на карте;

− g_type – тип геопространственных данных слоя;

− css_style – CartoCSS-стиль отображения слоя;

− render_type – определяет тип отрисовки слоя:

1) "0" – отрисовка растром на сервере;
2) "1" – отрисовка вектором на клиенте.

− style_mode – определяет тип стиля отображения слоя:

1) "0" – тип не задан;
2) "1" – простой стиль;
3) "2" – CartoCSS;
4) "3" – визуализация.

Вышеописанный ответ может отличаться в следующих случаях:

− если запрашиваемый слой является виртуальным, то изменится значение поля "type" с "*layer*" на "*virtual*";

− если запрашиваемый слой является визуализацией, то добавится поле "visualSettings", значением которого будет описывающий настройки визуализации объект формата json.

Работа с визуализациями описана в документе «Программное обеспечение «Ситроникс.ГИС». Руководство оператора».

Программа содержит восемь типов визуализаций:

- тематическая карта;
- категории;
- пузырьковые диаграммы;
- круговые диаграммы;
- столбчатые диаграммы;
- линейные диаграммы;
- радар;
- тепловая карта.

Тематическая карта


```
{
   "...": "...",
   "visualSettings": {
      "id": 16,
      "type": 1,
      "column": 5274,
      "columnCode": "population",
      "algorithm": 3,
      "classesCount": 9,
      "classesSettings": {
         "opacity": 80,
         "lineWidth": 1.0,
         "midColor": "#00bfaf",
         "linePattern": 1,
         "minColor": "#ffffcc",
         "maxColor": "#002080"
      },
```

"pointSettings": {
"size": 10.0,
"allowOverlap": true,
"sizeEnabled": false
},
"showBorder": true,
"borderSettings": {
"opacity": 100,
"pattern": 1,
"width": 1.0,
"color": "#FFFFFF"
},
"showText": true,
"textSettings": {
"fontSize": 12.0,
"halo": true,
"haloOpacity": 100,
"fontOpacity": 100,
"fontFamily": "Arial Bold",
"allowOverlap": false,
"haloWidth": 1.0,
"haloColor": "#ffffff",
"column": "gis_id",
"fontColor": "#000000",
"textOffset": 10.0,
"offsetEnabled": false
},
"showLegend": true,
"showLegendName": true,
"legendName": "Тематическая карта [mir]",

```
        "legendNames": [
          {
            "ru": "Тематическая карта [mir]"
          },
          {
            "fr": ""
          },
          {
            "en": ""
          }
        ],
        "showValues": false
      }
    }
```


где:

- id – уникальный идентификатор настроек визуализации;
- type – тип визуализации;
- column – уникальный идентификатор колонки, по значениям которой строится визуализация;
- columnCode – код колонки, по значениям которой строится визуализация;
- algorithm – алгоритм распределения:
    1) "1" – равные интервалы;
    2) "2" – квантили;
    3) "3" – естественные границы.
- classesCount – количество классов;
- classesSettings – настройки отображения классов:
    1) opacity – непрозрачность;
    2) lineWidth – ширина линии;
    3) midColor – промежуточный цвет;

4) linePattern – тип линии;
5) minColor – начальный цвет;
6) maxColor – конечный цвет.

- pointSettings – настройки отображения точки:
  1) size – размер точки;
  2) allowOverlap – перекрытие объектов;
  3) sizeEnabled – признак управления размером точки.
- showBorder – признак отображения контура;
- borderSettings – настройки отображения контура:
  1) opacity – непрозрачность;
  2) pattern – тип отображения;
  3) width – ширина;
  4) color – цвет.
- showText – признак отображения текста;
- textSettings – настройки отображения текста:
  1) fontSize – размер текста;
  2) halo – признак отображения контура текста;
  3) haloOpacity – непрозрачность контура текста;
  4) fontOpacity – непрозрачность текста;
  5) fontFamily – шрифт текста;
  6) allowOverlap – признак перекрытия текста;
  7) haloWidth – ширина контура текста;
  8) haloColor – цвет контура текста;
  9) column – код колонки, значения которой будут отображаться;
  10) fontColor – цвет текста;
  11) textOffset – смещение текста;
  12) offsetEnabled – признак смещения текста.
- showLegend – признак отображения легенды;
- showLegendName – признак отображения названия легенды;

- legendName – название легенды;
- legendNames – список локализованных значений названия легенды;
- showValues – признак отображения значений.

Категории


```
{
   "...": "...",
   "visualSettings": {
      "id": 17,
      "type": 2,
      "column": 5277,
      "columnCode": "gis_id",
      "algorithm": 1,
      "classesCount": 9,
      "classesSettings": {
       "opacity": 80,
       "linePattern": 1,
       "defaultClass": {
         "count": 46,
         "value": "Остальные",
         "icon": null,
         "checked": true,
         "useIcon": false,
         "color": "#05cfba"
       },
       "lineWidth": 1.0,
       "classes": [
         {
           "count": 1,
           "value": 2,
```

"icon": null,
"checked": true,
"useIcon": false,
"color": "#80f320"
},
{
"count": 1,
"value": 3,
"icon": null,
"checked": true,
"useIcon": false,
"color": "#86f11c"
},
{
"...": "..."
},
{
"count": 1,
"value": 100,
"icon": null,
"checked": true,
"useIcon": false,
"color": "#02c4c5"
},
{
"count": 1,
"value": 101,
"icon": null,
"checked": true,
"useIcon": false,

```
      "color": "#03cac0"
    }
  ]
},
"pointSettings": {
  "sizeEnabled": false,
  "allowOverlap": true,
  "size": 10.0
},
"showBorder": true,
"borderSettings": {
  "opacity": 100,
  "pattern": 1,
  "width": 1.0,
  "color": "#FFFFFF"
},
"showText": false,
"textSettings": {
  "fontSize": 12.0,
  "haloOpacity": 100,
  "fontOpacity": 100,
  "fontFamily": "Arial Bold",
  "haloWidth": 1.0,
  "haloColor": "#ffffff",
  "textOffset": 10.0,
  "column": "gis_id",
  "halo": true,
  "allowOverlap": false,
  "fontColor": "#000000",
  "offsetEnabled": false
```

```
        },
        "showLegend": false,
        "showLegendName": true,
        "legendName": "Категории [mir]",
        "legendNames": [
         {
          "ru": "Категории [mir]"
         },
         {
          "fr": ""
         },
         {
          "en": ""
         }
        ],
        "showValues": false
      }
    }
```


где:

- id – уникальный идентификатор настроек визуализации;
- type – тип визуализации;
- column – уникальный идентификатор колонки, по значениям которой строится визуализация;
- columnCode – код колонки, по значениям которой строится визуализация;
- algorithm – алгоритм распределения. Для данного типа визуализации всегда имеет "1";
- classesCount – количество классов;
- classesSettings – настройки отображения классов:
    1) opacity – непрозрачность;

2) linePattern – тип линии;
3) defaultClass – настройки отображения класса по умолчанию:
    1) count – количество объектов, к которым применены текущие настройки отображения;
    2) value – значение;
    3) icon – иконка;
    4) checked – признак отображения объектов этого класса;
    5) useIcon – признак отображения иконки;
    6) color – цвет.
4) lineWidth – ширина линии;
5) classes – список настроек отображения классов. Структура каждого элемента соответствует структуре поля "defaultClass".

- pointSettings – настройки отображения точки:
    1) size – размер точки;
    2) allowOverlap – перекрытие объектов;
    3) sizeEnabled – признак управления размером точки.
- showBorder – признак отображения контура;
- borderSettings – настройки отображения контура:
    1) opacity – непрозрачность;
    2) pattern – тип отображения;
    3) width – ширина;
    4) color – цвет.
- showText – признак отображения текста;
- textSettings – настройки отображения текста:
    1) fontSize – размер текста;
    2) halo – признак отображения контура текста;
    3) haloOpacity – непрозрачность контура текста;
    4) fontOpacity – непрозрачность текста;
    5) fontFamily – шрифт текста;

6) allowOverlap – признак перекрытия текста;
7) haloWidth – ширина контура текста;
8) haloColor – цвет контура текста;
9) column – код колонки, значения которой будут отображаться;
10) fontColor – цвет текста;
11) textOffset – смещение текста;
12) offsetEnabled – признак смещения текста.

- showLegend – признак отображения легенды;
- showLegendName – признак отображения названия легенды;
- legendName – название легенды;
- legendNames – список локализованных значений названия легенды;
- showValues – признак отображения значений.

Пузырьковые диаграммы


```
{
  "...": "...",
  "visualSettings": {
    "id": 18,
    "type": 3,
    "column": 5281,
    "columnCode": "gis_id",
    "algorithm": 1,
    "classesCount": 9,
    "classesSettings": {
      "pointSizeMax": 50.0,
      "icon": null,
      "opacity": 80,
      "pointSizeMin": 10.0,
      "useIcon": false,
```

"color": "#ff6666"
},
"pointSettings": {
"size": 10.0,
"allowOverlap": true,
"sizeEnabled": false
},
"showBorder": true,
"borderSettings": {
"opacity": 100,
"pattern": 1,
"width": 1.0,
"color": "#FFFFFF"
},
"showText": false,
"textSettings": {
"fontSize": 12.0,
"halo": true,
"haloOpacity": 100,
"fontOpacity": 100,
"fontFamily": "Arial Bold",
"allowOverlap": false,
"haloWidth": 1.0,
"haloColor": "#ffffff",
"column": "gis_id",
"fontColor": "#000000",
"textOffset": 10.0,
"offsetEnabled": false
},
"showLegend": false,

"showLegendName": true,
"legendName": "Пузырьковая диаграмма [mir]",
"legendNames": [
{
"ru": "Пузырьковая диаграмма [mir]"
},
{
"fr": ""
},
{
"en": ""
}
],
"showValues": false
}
}

где:

- id – уникальный идентификатор настроек визуализации;
- type – тип визуализации;
- column – уникальный идентификатор колонки, по значениям которой строится визуализация;
- columnCode – код колонки, по значениям которой строится визуализация;
- algorithm – алгоритм распределения:
    1) "1" – равные интервалы;
    2) "2" – квантили;
    3) "3" – естественные границы.
- classesCount – количество классов;
- classesSettings – настройки отображения классов:
    1) pointSizeMax – максимальный размер маркера;

2) icon – иконка;
3) opacity – непрозрачность;
4) pointSizeMin – минимальный размер маркера;
5) useIcon – признак отображения иконки;
6) color – цвет.

- pointSettings – настройки отображения точки:
  1) size – размер точки;
  2) allowOverlap – перекрытие объектов;
  3) sizeEnabled – признак управления размером точки.
- showBorder – признак отображения контура;
- borderSettings – настройки отображения контура:
  1) opacity – непрозрачность;
  2) pattern – тип отображения;
  3) width – ширина;
  4) color – цвет.
- showText – признак отображения текста;
- textSettings – настройки отображения текста:
  1) fontSize – размер текста;
  2) halo – признак отображения контура текста;
  3) haloOpacity – непрозрачность контура текста;
  4) fontOpacity – непрозрачность текста;
  5) fontFamily – шрифт текста;
  6) allowOverlap – признак перекрытия текста;
  7) haloWidth – ширина контура текста;
  8) haloColor – цвет контура текста;
  9) column – код колонки, значения которой будут отображаться;
  10) fontColor – цвет текста;
  11) textOffset – смещение текста;
  12) offsetEnabled – признак смещения текста.

- showLegend – признак отображения легенды;
- showLegendName – признак отображения названия легенды;
- legendName – название легенды;
- legendNames – список локализованных значений названия легенды;
- showValues – признак отображения значений.

Круговые диаграммы


```
{
   "...": "...",
   "visualSettings": {
      "id": 19,
      "type": 4,
      "column": 5285,
      "columnCode": Op",
      "algorithm": 1,
      "classesCount": 9,
      "classesSettings": {
         "sizeFrom": 60,
         "showScale": false,
         "sizeColumnCode": "gis_id",
         "sizeTo": 200,
         "allowOverlap": false,
         "opacity": 80,
         "columns": [
            {
               "code": "gis_id",
               "id": 5285,
               "color": "#d4b301"
            }
         ],
```

"sizeValue": 100,
"sizeType": 1,
"sizeColumn": 5285
},
"pointSettings": {
"size": 10.0,
"allowOverlap": true,
"sizeEnabled": false
},
"showBorder": true,
"borderSettings": {
"opacity": 100,
"pattern": 1,
"width": 1.0,
"color": "#FFFFFF"
},
"showText": false,
"textSettings": {
"fontSize": 12.0,
"halo": true,
"haloOpacity": 100,
"fontOpacity": 100,
"fontFamily": "Arial Bold",
"allowOverlap": false,
"haloWidth": 1.0,
"haloColor": "#ffffff",
"column": "gis_id",
"fontColor": "#000000",
"textOffset": 10.0,
"offsetEnabled": false

},
"showLegend": false,
"showLegendName": true,
"legendName": "Круговые диаграммы [mir]",
"legendNames": [
{
"ru": "Круговые диаграммы [mir]"
},
{
"fr": ""
},
{
"en": ""
}
],
"showValues": false
}
}

где:

- id – уникальный идентификатор настроек визуализации;
- type – тип визуализации;
- column – уникальный идентификатор колонки, по значениям которой строится визуализация;
- columnCode – код колонки, по значениям которой строится визуализация;
- algorithm – алгоритм распределения. Для данного типа визуализации всегда имеет "1";
- classesCount – количество классов;
- classesSettings – настройки отображения классов:
    1) sizeFrom – размер от …;

2) showScale – признак отображения шкалы;
3) sizeColumnCode – код колонки, по которой считается размер;
4) sizeTo – размер до ...;
5) allowOverlap – признак перекрытия;
6) opacity – непрозрачность;
7) columns – список колонок, по которым строится визуализация:
    1) code – код колонки;
    2) id – уникальный идентификатор колонки;
    3) color – цвет.
8) sizeValue – значение размера;
9) sizeType – тип размера:
    1) "1" – фиксированный;
    2) "2" – сумма значений;
    3) "3" – по колонке.
10) sizeColumn – уникальный идентификатор колонки, по которой считается размер.

- pointSettings – настройки отображения точки:
    1) size – размер точки;
    2) allowOverlap – перекрытие объектов;
    3) sizeEnabled – признак управления размером точки.
- showBorder – признак отображения контура;
- borderSettings – настройки отображения контура:
    1) opacity – непрозрачность;
    2) pattern – тип отображения;
    3) width – ширина;
    4) color – цвет.
- showText – признак отображения текста;
- textSettings – настройки отображения текста:
    1) fontSize – размер текста;

2) halo – признак отображения контура текста;
3) haloOpacity – непрозрачность контура текста;
4) fontOpacity – непрозрачность текста;
5) fontFamily – шрифт текста;
6) allowOverlap – признак перекрытия текста;
7) haloWidth – ширина контура текста;
8) haloColor – цвет контура текста;
9) column – код колонки, значения которой будут отображаться;
10) fontColor – цвет текста;
11) textOffset – смещение текста;
12) offsetEnabled – признак смещения текста.

- showLegend – признак отображения легенды;
- showLegendName – признак отображения названия легенды;
- legendName – название легенды;
- legendNames – список локализованных значений названия легенды;
- showValues – признак отображения значений.

Столбчатые диаграммы

```
{
   "...": "...",
   "visualSettings": {
      "id": 20,
      "type": 5,
      "column": 5289,
      "columnCode": "gis_id",
      "algorithm": 1,
      "classesCount": 9,
      "classesSettings": {
         "sizeFrom": 60,
```

```
    "showScale": true,
    "sizeColumnCode": "gis_id",
    "sizeTo": 200,
    "allowOverlap": false,
    "opacity": 80,
    "columns": [
      {
        "code": "gis_id",
        "id": 5289,
        "label": "gis_id",
        "color": "#fe3f38"
      }
    ],
    "sizeValue": 60,
    "sizeType": 1,
    "sizeColumn": 5289
  },
  "pointSettings": {
    "size": 10.0,
    "allowOverlap": true,
    "sizeEnabled": false
  },
  "showBorder": true,
  "borderSettings": {
    "opacity": 100,
    "pattern": 1,
    "width": 1.0,
    "color": "#FFFFFF"
  },
  "showText": false,
```

```
  "textSettings": {
    "fontSize": 12.0,
    "halo": true,
    "haloOpacity": 100,
    "fontOpacity": 100,
    "fontFamily": "Arial Bold",
    "allowOverlap": false,
    "haloWidth": 1.0,
    "haloColor": "#ffffff",
    "column": "gis_id",
    "fontColor": "#000000",
    "textOffset": 10.0,
    "offsetEnabled": false
  },
  "showLegend": false,
  "showLegendName": true,
  "legendName": "Столбчатые диаграммы [mir]",
  "legendNames": [
    {
      "ru": "Столбчатые диаграммы [mir]"
    },
    {
      "fr": ""
    },
    {
      "en": ""
    }
  ],
  "showValues": false
}
```

}

где:

- id – уникальный идентификатор настроек визуализации;
- type – тип визуализации;
- column – уникальный идентификатор колонки, по значениям которой строится визуализация;
- columnCode – код колонки, по значениям которой строится визуализация;
- algorithm – алгоритм распределения. Для данного типа визуализации всегда имеет "1";
- classesCount –количество классов;
- classesSettings – настройки отображения классов:
    1) sizeFrom – размер от …;
    2) showScale – признак отображения шкалы;
    3) sizeColumnCode – код колонки, по которой считается размер;
    4) sizeTo – размер до ...;
    5) allowOverlap – признак перекрытия;
    6) opacity – непрозрачность;
    7) columns – список колонок, по которым строится визуализация:
        1) code – код колонки;
        2) id – уникальный идентификатор колонки;
        3) label – название колонки;
        4) color – цвет.
    8) sizeValue – значение размера;
    9) sizeType – тип размера:
        1) 1 – фиксированный;
        2) 2 – сумма значений;
        3) 3 – по колонке.

10) sizeColumn – уникальный идентификатор колонки, по которой считается размер.

- pointSettings – настройки отображения точки:
    1) size – размер точки;
    2) allowOverlap – перекрытие объектов;
    3) sizeEnabled – признак управления размером точки;
- showBorder – признак отображения контура;
- borderSettings – настройки отображения контура:
    1) opacity – непрозрачность;
    2) pattern – тип отображения;
    3) width – ширина;
    4) color – цвет.
- showText – признак отображения текста;
- textSettings – настройки отображения текста:
    1) fontSize – размер текста;
    2) halo – признак отображения контура текста;
    3) haloOpacity – непрозрачность контура текста;
    4) fontOpacity – непрозрачность текста;
    5) fontFamily – шрифт текста;
    6) allowOverlap – признак перекрытия текста;
    7) haloWidth – ширина контура текста;
    8) haloColor – цвет контура текста;
    9) column – код колонки, значения которой будут отображаться;
    10) fontColor – цвет текста;
    11) textOffset – смещение текста;
    12) offsetEnabled – признак смещения текст.
- showLegend – признак отображения легенды;
- showLegendName – признак отображения названия легенды;
- legendName – название легенды;

- legendNames – список локализованных значений названия легенды;
- showValues – признак отображения значений.

Линейные диаграммы


```
{
  "...": "...",
  "visualSettings": {
    "id": 21,
    "type": 6,
    "column": 5293,
    "columnCode": "gis_id",
    "algorithm": 1,
    "classesCount": 9,
    "classesSettings": {
      "sizeFrom": 60,
      "showScale": true,
      "sizeColumnCode": "gis_id",
      "sizeTo": 200,
      "allowOverlap": false,
      "opacity": 80,
      "columns": [
        {
          "code": "population",
          "id": 5296,
          "color": "#dc0a86"
        }
      ],
      "sizeValue": 60,
      "sizeType": 1,
      "sizeColumn": 5293
```

},
"pointSettings": {
"size": 10.0,
"allowOverlap": true,
"sizeEnabled": false
},
"showBorder": true,
"borderSettings": {
"opacity": 100,
"pattern": 1,
"width": 1.0,
"color": "#FFFFFF"
},
"showText": false,
"textSettings": {
"fontSize": 12.0,
"halo": true,
"haloOpacity": 100,
"fontOpacity": 100,
"fontFamily": "Arial Bold",
"allowOverlap": false,
"haloWidth": 1.0,
"haloColor": "#ffffff",
"column": "gis_id",
"fontColor": "#000000",
"textOffset": 10.0,
"offsetEnabled": false
},
"showLegend": false,
"showLegendName": true,

"legendName": "Линейные диаграммы [mir]",
"legendNames": [
{
"ru": "Линейные диаграммы [mir]"
},
{
"fr": ""
},
{
"en": ""
}
],
"showValues": false
}
}

где:

- id – уникальный идентификатор настроек визуализации;
- type – тип визуализации;
- column – уникальный идентификатор колонки, по значениям которой строится визуализация;
- columnCode – код колонки, по значениям которой строится визуализация;
- algorithm – алгоритм распределения. Для данного типа визуализации всегда имеет "1";
- classesCount – количество классов;
- classesSettings – настройки отображения классов:
    1) sizeFrom – размер от …;
    2) showScale – признак отображения шкалы;
    3) sizeColumnCode – код колонки, по которой считается размер;
    4) sizeTo – размер до …;

5) allowOverlap – признак перекрытия;
6) opacity – непрозрачность;
7) columns – список колонок по которым строится визуализация:
    1) code – код колонки;
    2) id – уникальный идентификатор колонки;
    3) color – цвет.
8) sizeValue – значение размера;
9) sizeType – тип размера:
    1) 1 – фиксированный;
    2) 2 – сумма значений;
    3) 3 – по колонке.
10) sizeColumn – уникальный идентификатор колонки, по которой считается размер.

- pointSettings – настройки отображения точки:
    1) size – размер точки;
    2) allowOverlap – перекрытие объектов;
    3) sizeEnabled – признак управления размером точки.
- showBorder – признак отображения контура;
- borderSettings – настройки отображения контура:
    1) opacity – непрозрачность;
    2) pattern – тип отображения;
    3) width – ширина;
    4) color – цвет.
- showText – признак отображения текста;
- textSettings – настройки отображения текста:
    1) fontSize – размер текста;
    2) halo – признак отображения контура текста;
    3) haloOpacity – непрозрачность контура текста;
    4) fontOpacity – непрозрачность текста;

5) fontFamily – шрифт текста;
6) allowOverlap – признак перекрытия текста;
7) haloWidth – ширина контура текста;
8) haloColor – цвет контура текста;
9) column – код колонки, значения которой будут отображаться;
10) fontColor – цвет текста;
11) textOffset – смещение текста;
12) offsetEnabled – признак смещения текста.

- showLegend – признак отображения легенды;
- showLegendName – признак отображения названия легенды;
- legendName – название легенды;
- legendNames – список локализованных значений названия легенды;
- showValues – признак отображения значений.

Тепловая карта


```
{
  "...": "...",
  "visualSettings": {
    "id": 22,
    "type": 7,
    "column": 5297,
    "columnCode": "gis_id",
    "algorithm": 1,
    "classesCount": 9,
    "classesSettings": {
      "markerSizeMin": 60,
      "layerOpacity": 80,
      "weightNormalizeMax": 10,
      "useWeight": false,
```

```
    "markerSizeType": "fixed",
    "linkToScale": false,
    "markerSizeMax": 200,
    "weightNormalizeMin": 0.1,
    "markerOpacity": 80,
    "markerSize": 35.0,
    "useWeightField": 5297,
    "markerSizeField": 5297,
    "linkToScaleZoom": 10,
    "colors": [
        "#0000ff",
        "#00ffff",
        "#90EE90",
        "#ffff00",
        "#ffa500",
        "#ff0000"
    ],
    "weightNormalize": false
},
"pointSettings": {
    "size": 10.0,
    "allowOverlap": true,
    "sizeEnabled": false
},
"showBorder": true,
"borderSettings": {
    "opacity": 100,
    "pattern": 1,
    "width": 1.0,
    "color": "#FFFFFF"
```

},
"showText": false,
"textSettings": {
"fontSize": 12.0,
"halo": true,
"haloOpacity": 100,
"fontOpacity": 100,
"fontFamily": "Arial Bold",
"allowOverlap": false,
"haloWidth": 1.0,
"haloColor": "#ffffff",
"column": "gis_id",
"fontColor": "#000000",
"textOffset": 10.0,
"offsetEnabled": false
},
"showLegend": false,
"showLegendName": true,
"legendName": "Тепловая карта [mir]",
"legendNames": [
{
"ru": "Тепловая карта [mir]"
},
{
"fr": ""
},
{
"en": ""
}
],

```
        "showValues": false
    }
}
```

где:

- id – уникальный идентификатор настроек визуализации;
- type – тип визуализации;
- column – уникальный идентификатор колонки, по значениям которой строится визуализация;
- columnCode – код колонки, по значениям которой строится визуализация;
- algorithm – алгоритм распределения. Для данного типа визуализации всегда имеет "1";
- classesCount – количество классов;
- classesSettings – настройки отображения классов:
    1) markerSizeMin – минимальный размер маркера;
    2) layerOpacity – непрозрачность;
    3) weightNormalizeMax – максимальное значение нормализации веса;
    4) useWeight – признак использования нормализации по весу;
    5) markerSizeType – тип радиуса влияния маркера:
        1) **fixed** – фиксированный;
        2) **field** – по колонке.
    6) linkToScale – признак привязки к масштабу;
    7) markerSizeMax – максимальный размер маркера;
    8) weightNormalizeMin – минимальное значение нормализации веса;
    9) markerOpacity – интенсивность;
    10) markerSize – размер маркера;
    11) useWeightField – уникальный идентификатор колонки, значения которой нормализуются по весу;

12) markerSizeField – уникальный идентификатор колонки, по значениям которой вычисляется радиус влияния;
13) linkToScaleZoom – значение привязки зума;
14) colors – список значений цветов, в градиенте которых меняется окраска маркера;
15) weightNormalize – признак нормализации по весу.

- pointSettings – настройки отображения точки:
  1) size – размер точки;
  2) allowOverlap – перекрытие объектов;
  3) sizeEnabled – признак управления размером точки.
- showBorder – признак отображения контура;
- borderSettings – настройки отображения контура:
  1) opacity – непрозрачность;
  2) pattern – тип отображения;
  3) width – ширина;
  4) color – цвет.
- showText – признак отображения текста;
- textSettings – настройки отображения текста:
  1) fontSize – размер текста;
  2) halo – признак отображения контура текста;
  3) haloOpacity – непрозрачность контура текста;
  4) fontOpacity – непрозрачность текста;
  5) fontFamily – шрифт текста;
  6) allowOverlap – признак перекрытия текста;
  7) haloWidth – ширина контура текста;
  8) haloColor – цвет контура текста;
  9) column – код колонки, значения которой будут отображаться;
  10) fontColor – цвет текста;
  11) textOffset – смещение текста;

12) offsetEnabled – признак смещения текста.

- showLegend – признак отображения легенды;
- showLegendName – признак отображения названия легенды;
- legendName – название легенды;
- legendNames – список локализованных значений названия легенды;
- showValues – признак отображения значений.

Если включен режим отображения слоя вектором, то добавляется поле "vectorSettings", значением которого будет объект формата JSON, описывающий стили отрисовки векторного слоя на клиенте.

Пример возможного значения:


```
{
   "...": "...",
   "vectorSettings": {
      "stylesPolygon": {},
      "stylesText": {},
      "minZoom": 2,
      "clusterSettings": {
         "objectsCountDx": 0,
         "icon": null,
         "showObjectsCount": true,
         "iconSizeMax": 25,
         "radius": 40,
         "iconSizeEnabled": false,
         "iconDx": 0,
         "iconEnabled": false,
         "iconOffsetEnabled": false,
         "iconDy": 0,
         "iconSizeMin": 15,
         "clusteringType": "client",
```

"objectsCountDy": 0,
"maxZoom": 12,
"maxZoomEnabled": false,
"splitByCategories": false
},
"popupColumn": 1065,
"maxZoom": 5,
"popupColumnCode": "gis_id",
"showPopup": false,
"zoomsEnabled": false,
"clusteringEnabled": true,
"id": 21,
"stylesPoint": {
"icon": null,
"iconDx": 0,
"iconOffsetEnabled": true,
"iconDy": 0
},
"stylesLine": {}
}
}

где:

- stylesPolygon – не используются в настоящий момент;
- stylesText – не используются в настоящий момент;
- minZoom – минимальное значение ограничения по зуму;
- clusterSettings – настройки кластеризации:
    1) objectsCountDx – смещение количества объектов по оси X;
    2) icon – иконка;
    3) showObjectsCount – признак отображения количества объектов;

4) iconSizeMax – максимальный размер иконки;
5) radius – радиус кластеризации;
6) iconSizeEnabled – признак использования ограничений размеров иконки;
7) iconDx – смещение иконки кластера по оси X;
8) iconEnabled – признак использования иконки;
9) iconOffsetEnabled – признак смещения иконки;
10) iconDy – смещение иконки по оси Y;
11) iconSizeMin – минимальный размер иконки;
12) clusteringType – тип кластеризации;
13) objectsCountDy – смещение количества объектов по оси Y;
14) maxZoom – максимальный зум кластеризации;
15) maxZoomEnabled – признак использования максимального зума;
16) splitByCategories – флаг, отвечающий за способ формирования кластеров:
    1) true – в кластеры будут попадать только объекты одинаковых категорий;
    2) false – в кластеры будут попадать все объекты (без учета категории).

− popupColumn м уникальный идентификатор колонки, значение которой будет использоваться в сплывающей подсказке;

− maxZoom м максимальное значение ограничения по зуму;

− popupColumnCode – код колонки, значение которой будет использоваться в сплывающей подсказке;

− showPopup – признак использования всплывающей подсказки;

− zoomsEnabled – признак ограничения по зуму;

− clusteringEnabled – признак использования кластеризации;

− id – уникальный идентификатор настроек отображения;

− stylesPoint – настройки отображения иконки:

1) icon – иконка;
2) iconDx – смещение иконки по оси X;
3) iconOffsetEnabled – признак использования смещения колонки;
4) iconDy – смещение иконки по оси Y.

− stylesLine – параметр не используется.

3.1.4.3 Изменение слоя карты

Для изменения слоя используется метод PUT, если необходимо изменить объект слоя полностью, или PATCH, если необходимо изменить отдельные поля.

Запрос:

PUT/PATCH http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/

Для выполнения запроса необходим токен авторизации.

Параметры создаваемого слоя в формате JSON должны передаваться в теле запроса:

```
{
   "code": "new_code",
   "localized_name": false,
   "name": "Имя",
   "parent_id": 2,
   "sort": 3.5,
   "css_style": "CartoCSS-стиль отображения слоя"
}
```

где:

− code (необязательный) – новый код слоя. Если не указан, то значение останется прежним;

− localized_name (необязательный) – флаг, локализовать ли название слоя. Если не указан, то значение останется прежним;

− name (необязательный) – имя слоя или массив локализованных названий слоя в формате *{"ru": "Новый слой", "en": "New Layer"}*, где "ru" и "en" – коды языков локализации (если выставлен флаг "localized_name");

− parent_id (обязательный) – идентификатор родительского контейнера;

− sort (необязательный) – значение сортировки слоя. Если не указан, то значение останется прежним;

− css_style (необязательный) – CartoCSS-стиль отображения слоя. Если не указан, то значение останется прежним.

Ответ:

Структура ответа на запрос изменения слоя соответствует ответу на запрос получения объекта слоя карты.

Если не указан идентификатор родительского контейнера, указан несуществующий, или родительский контейнер принадлежит другой карте, то сервер вернет ошибку со статусом 404.

Если указан не валидный CartoCSS-стиль отображения слоя, то сервер вернет ошибку со статусом 500.

Описание ошибок приведено в п. 3.1.12.

3.1.4.4 Удаление слоя карты

Запрос:

DELETE http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/

Для выполнения запроса необходим токен авторизации.

Ответ: true.

Если попытаться удалить несуществующий слой, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.4.5 Получение легенды слоя карты

Получить легенду можно только для слоя с типом Визуализация.

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/legend/

Ответ:

{
"name": "Название легенды",
"column": "gis_id",
"showValues": true,
"showLegend": true,
"minValue": 1,
"maxValue": 200,
"type": 5,
"classificators": [
{
"value": "gis_id"
,
, "fill":"#f82851",
"isIcon": false
}
],
}

где:

- name – заголовок легенды;
- column – код поля, из которого берутся значения для визуализации;
- showValues – признак, будут ли отображаться граничные значения;
- showLegend – признак, отображается ли легенда;
- minValue – минимальное значение шкалы легенды;
- maxValue – максимальное значение шкалы легенды;

− type – тип визуализации;

− classificators – список стилей отображения для каждой группы объектов визуализации.

Если запрошен несуществующий слой или запрошен слой с типом отличным от Визуализация, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.4.6 Получение габарита слоя карты

Получить габарит можно только для слоёв и виртуальных слоёв.

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/bbox/

Необязательные параметры запроса:

− bboxSR – spatial reference для габаритов объекта в формате EPSG. Возможные значения: 4326 (значение по умолчанию) и 3857.

Ответ:

```
{
   "sw" : [Lat, Lon],
   "ne" : [Lat, Lon]
}
```

где: bbox – значение габарита слоя в проекции EPSG:bboxSR.

Если запрошен несуществующий слой или запрошен контейнер, не являющийся слоем (например папка), то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.4.7 Обновление материализованного слоя

Для выполнения запроса необходим токен авторизации.

Запрос:

POST http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/refresh/

Ответ:

Структура ответа на запрос обновления материализованного слоя соответствует ответу на запрос получения объекта слоя карты.

Если запрос выполняется с токеном авторизации пользователя, не являющимся владельцем материализованного слоя, то сервер вернет ошибку со статусом 403.

Если обновляемый слой не является материализованным, то сервер вернет ошибку со статусом 500.

Описание ошибок приведено в п. 3.1.12.

3.1.5 Работа со структурой слоя карты

В базе данных слой соответствует таблице. Любой слой имеет, как минимум, два поля:

- gis_id – уникальный идентификатор объекта;
- geom – геопространственные данные объекта.

Дополнительно слой может содержать достаточно большое количество пользовательских полей. Максимальное количество зависит от типа создаваемых полей и может достигать 1000 штук.

Код пользовательских полей:

- может состоять из букв латинского алфавита, цифр от 0 до 9 или знака подчеркивания;
- не может начинаться с цифры;
- не должен превышать 44 символов.

Возможные типы полей, и их соответствие типам в базе данных:

- number – число. В БД имеет тип "float8". Исключением в данном случае будет служебная колонка "gis_id" – в БД имеет тип "int4";

− geom – геопространственные данные. В БД имеет пользовательский тип "Geometry";

− string – строка. В БД имеет тип "varchar";

− text – текст. В БД имеет тип "text";

− datetime – дата. В БД имеет тип "datetime";

− bool – логический. В БД имеет тип "bool";

− file – файл. В БД имеет тип "text". В одном файловом поле объекта может храниться неограниченное число файлов, т. к. они хранятся в виде текстовой строки, которая состоит из первичных ключей файлов, разделенных запятой;

− title – заголовок. Не имеет отдельной колонки в БД, т. к. является логической абстракцией и используется для объединения колонок общим заголовком.

3.1.5.1 Получение структуры слоя

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/structure/

Ответ:


```
[
   {
      "id":48,
      "sort":1,
      "parent_id":0,
      "code":"gis_id",
      "name":"",
      "required": true,
      "type":"number",
      "localized_name":false,
      "localized_value":false
```

},
{
"id":49,
"sort":2,
"parent_id":0,
"code":"geom",
"name":"",
"required": false,
"type":"geom",
"localized_name":false,
"localized_value":false
},
{
"id":50,
"sort":3,
"parent_id":0,
"code":"titles",
"name":"Подписи",
"type":"title",
"localized_name":true,
"localized_value":false
},
{
"id":51,
"sort":1,
"parent_id":50,
"code":"label_en",
"name":"Подпись английская",
"required": true,
"type":"title|number|string|text|datetime|bool|file|geom",

"localized_name":true,
"localized_value":false
},
{
"id":52,
"sort":2,
"parent_id":50,
"code":"label_ru",
"name":"Подпись русская",
"required": false,
"type":"title|number|string|text|datetime|bool|file|geom",
"localized_name":true,
"localized_value":false
}
]

Каждый элемент списка описывает одно поле слоя:

- id – уникальный идентификатор поля в пределах проекта;
- sort – порядок сортировки в пределах родителя (поля "parent_id");
- parent_id – идентификатор родительского поля, используется для построения древовидной структуры полей, например, несколько полей объединены общим заголовком;
- code – код поля слоя. Уникальный в пределах слоя. Соответствует коду колонки таблицы в БД;
- name – название поля слоя. Может не иметь значения;
- required – признак заполнения поля при сохранении объекта;
- type – тип поля;
- localized_name – признак локализации имени поля;
- localized_value – признак локализации значения поля.

Если запрошен несуществующий слой, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.5.2 Создание колонки

Запрос:

POST http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/structure/

Для выполнения запроса необходим токен авторизации. Параметры создаваемой колонки в формате JSON должны передаваться в теле запроса:

```
{
   "code": "New_column_1",
   "name": {"ru": "Новая колонка", "en": "New Column"},
   "localized_name": true,
   "localized_value": false,
   "required": false,
   "type": "number",
   "sort":4.5,
   "parent_id": 50,
   "...": "..."
}
```

где:

− sort (необязательный) – значение сортировки колонки. Если не указан, то колонка добавится последней;

− parent_id (необязательный) – идентификатор родительской колонки. Если не указан, то колонка добавится на верхний уровень, может быть равен только идентификатору колонки типа "заголовок";

− code (обязательный) – код колонки;

− name (необязательный) – имя колонки или массив локализованных названий колонки в формате *{"ru": "Новая колонка ru", "en": "New Column en"}*, где "ru" и "en" – коды языков локализации (при наличии установленного флага" localized_name");

− required (необязательный) – признак, должно ли быть заполнено поле при сохранении объекта. Если не указан, будет выставлено значение по умолчанию (False);

− type (обязательный) – тип колонки;

− localized_name (необязательный) – признак локализации имени колонки. Если не указан, будет выставлено значение по умолчанию (False);

− localized_value (необязательный) – признак локализации значения колонки. Если не указан, будет выставлено значение по умолчанию (False);

− format (необязательный) – формат значений объектов слоя для данной колонки. Форматирование значений выполняется в клиентском ПО Программы.

Ответ:

Структура ответа на запрос изменения колонки слоя соответствует структуре элемента в ответе на запрос «Получение структуры слоя».

Если не указан тип поля, то сервер вернет ошибку со статусом 500. Описание ошибок приведено в п. 3.1.12.

3.1.5.3 Изменение колонки

Для изменения колонки слоя используется метод PUT, если необходимо изменить все поля колонки, или PATCH, если необходимо изменить отдельные поля.

Запрос:

PUT/PATCH http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/structure/<field_code>/

Для выполнения запроса необходим токен авторизации. Параметры изменяемой колонки в json-формате должны передаваться в теле запроса:

```
{
   "code": "New_column_1",
   "name": "Новая колонка 1",
```

```
    "localized_name": false,
    "localized_value": false,
    "required": false,
    "type": "number",
    "sort":4.5,
    "parent_id": 50,
    "...": "..."
}
```

где:

− sort (необязательный) – значение сортировки колонки. Если не указан, то значение останется прежним;

− parent_id (необязательный) – идентификатор родительской колонки. Если не указан, то значение останется прежним;

− code (необязательный) – новый код колонки. Если не указан, то значение останется прежним;

− name (необязательный) – имя колонки или массив локализованных названий колонки в формате {"ru": "Новая колонка", "en": "New Column en"}, где ru и en – коды языков локализации (в случае если выставлен флаг localized_name);

− required (необязательный) – признак, должно ли быть заполнено поле при сохранении объекта. Если не указан, то значение останется прежним;

− type (обязательный) – тип колонки. Необходимо указать либо текущий тип – если тип поля менять не надо, или новый тип, если необходимо изменить. Если значение в объекте не может быть явно приведено к новому типу, например, строка (имеющая в составе буквы) к дате, то значение в объекте будет потеряно;

− localized_name (необязательный) – признак, локализовано ли имя колонки. Если не указан, то значение останется прежним;

− localized_value (необязательный) – признак, локализовано ли значение колонки. Если не указан, то значение останется прежним;

− format (необязательный) – формат значений объектов слоя для данной колонки.

Ответ:

Структура ответа на запрос изменения колонки слоя соответствует структуре элемента в ответе на запрос «Получение структуры слоя».

Если не указан тип поля, то сервер вернет ошибку со статусом 500. Описание ошибок приведено в п. 3.1.12.

3.1.5.4 Удаление колонки

Запрос:

DELETE http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/structure/<field_code>/

Для выполнения запроса необходим токен авторизации.

Ответ: true.

Если попытаться удалить несуществующую колонку, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.6 Работа с данными слоя карты

3.1.6.1 Загрузка данных в слой

Запрос:

POST http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/import/

Для выполнения запроса необходим токен авторизации. Загрузка данных поддерживает следующие форматы:

Вектор:

− kml;

− shp;

− mid/mif (Mapinfo);

− tab (Mapinfo);

- geojson;
- xls/xlsx;
- dxf;
- csv;
- sxf (КБ Панорама).

Растр:

- tif;
- geotif;
- png;
- jpg/jpeg.

Возможные способы загрузки:

- через POST – GeoJSON в теле POST-запроса;
- через файл (excel, shp, ...).

Обязательные параметры:

- GeoJSON в теле POST-запроса;

либо

- file – файл в параметре POST-запроса.

Дополнительные необязательные параметры:

- srs – id проекции (в формате epsg/wkt/proj4) загружаемого файла (если нужна); по умолчанию: "4326"(EPSG) (формат задаваемой проекции определяется сервером автоматически);
- action – как производить импорт:
    1) "1" – заменить;
    2) "2" (по умолчанию) – объединить. Доступно только для векторных слоёв.
- raster_bbox – bbox для растровых данных в формате: BOX(x_min y_min,x_max y_max); значения по умолчанию не имеет, т.е. если растр не имеет данных о геопривязке и не будет задан этот параметр, это приведет к ошибке импорта.

Ответ:

Структура ответа на запрос загрузки данных в слой соответствует ответу на запрос «Получение объекта слоя карты».

3.1.6.2 Выгрузка данных слоя

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/export/

Выгрузка данных поддерживает следующие форматы:

Вектор:

- kml, kmz: (format=kml|kmz);
- shp: (format=shp);
- mid/mif, tab (MapInfo): (format=mif|tab);
- GeoJSON: (format=geojson);
- xlsx (Excel): (format=xls);
- csv: (format=csv).

Растр:

- GeoTIFF: (format=geotif).

Обязательные GET-параметры:

- format: SHP | MIF | TAB | GEOJSON | KML | KMZ | XLS | GEOTIF | CSV.

Необязательные дополнительные GET-параметры:

- langs: ru, en, ... (коды языков через ","), если параметр пустой, будут выгружены все языки;
- zip – принудительная упаковка в ZIP одиночных файлов (влияет только на экспорт в форматы, состоящие из одного файла);
- feature_ids – список идентификаторов объектов слоя (через запятую), которые необходимо экспортировать (иначе экспортируется весь слой).

Дополнительные GET-параметры для экспорта растровых слоев GeoTIFF:

− "orig" – скачивать оригинальный снимок или обработанный (true/false). По умолчанию – true.

Дополнительные GET-параметры для экспорта CSV:

− encoding – кодировка, в которую будут преобразованы текстовые данные экспортируемого слоя (utf-8, cp1251, ...). По умолчанию – utf-8;

− dialect – диалект (формат разделителя, экранирования строк и переносов) CSV для экспорта. По умолчанию – ";" (точка с запятой).

Поддерживаемые диалекты: "comma" – разделитель запятая.

Ответ:

zip-архив (для типов shp, mif) или файл (для типов geojson, kml, kmz, xls, csv) с результатом.

Если указан параметр zip, то файлы любого типа (кроме kmz) будут сжаты в *.zip. Экспорт в kmz всегда возвращает файл *.kmz, вне зависимости от наличия параметра сжатия.

Если указан недопустимый формат, то сервер вернет ошибку со статусом 500. Описание ошибок приведено в п. 3.1.12.

3.1.7 Работа с объектами слоя карты

При работе с объектами слоя, в структуре которого есть локализованные по значению колонки, необходимо указывать код мета-колонки (одним из свойств параметра "fields") и код языка (параметр "lng"), для которого необходимо получить/изменить значение. Здесь и далее по тексту считается, что, при отсутствии кода языка в параметре "lng", будет использоваться язык по умолчанию.

3.1.7.1 Получение количества объектов слоя

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/count/

Необязательные параметры запроса:

− precise – условие точного вычисления количества объектов:

1) false (по умолчанию) – подсчет производится быстро на основании внутреннего оптимизатора базы данных, возможен неточный результат;

2) true – подсчет производится точно, при большом количестве данных возможен долгий ответ от сервера.

Ответ: 50.

3.1.7.2 Получение списка объектов слоя

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/

Необязательные параметры запроса:

− fields – список кодов колонок через запятую, значения которых также попадут в результат; если необходимо получить все колонки, то необходимо указать «fields=*»;

− fcode – код колонки для фильтрации. Колонка, по которой производится фильтрация, должна быть указана в параметре "fields", или сервер вернет ошибку со статусом 500. Описание ошибок приведено в п. 3.1.12;

− fval – значение фильтруемой колонки. Если не нашлось объектов, удовлетворяющих условию фильтрации, то вернется пустой список и статус ответа будет "200";

− offset – стартовое смещение выдачи объектов;

− limit – ограничение на количество выбранных объектов;

− sort – набор колонок, по которым будет выполнена сортировка списка. Значение представляется в формате "name,a|capital|id,d", где "a=ASC" (значение по умолчанию) – по возрастанию, "d=DESC" – по убыванию;

− fbbox – фильтр по BBOX в формате "SW_LAT,SW_LON,NE_LAT,NE_LON";

− returnBbox – условие включения в ответ габаритов объекта. "0" – не включать (значение по умолчанию); "1" – включить bbox в ответ;

− bboxSR – spatial reference для габаритов объекта в формате EPSG. Возможные значения: 4326 (значение по умолчанию) и 3857. Данный параметр учитывается и при фильтрации по BBOX;

− returnGeom – условие включения в ответ геопространственных данных объекта. Геопространственные данные в ответе будут возвращены в формате GeoJSON. Возможные значения: "0" – не включать (значение по умолчанию); "1" – включить геопространственные данные в ответ;

− geomSR – spatial reference для геопространственных данных объекта в формате EPSG. Возможные значения: 4326 (значение по умолчанию) и 3857;

− geomZoom – параметр, учитываемый при запросе геопространственных данных объекта. При указании текущего зума, геопространственные данные будут автоматически упрощены под указанный зум. Если параметр не задавать, то геопространственные данные будут в исходном виде – без упрощения;

− returnCentroid – если задан параметр "returnGeom", то вместо геопространственных данных объектов будут возвращаться координаты точек их центроидов. Работает для всех типов геопространственных данных;

− lng – код языка, для которого будут получены локализованные значения в полях свойства "fields". Если параметр не указан, то будут получены значения для языка по умолчанию.

Ответ:

Каждый элемент ответа на запрос получения списков объектов слоя по структуре соответствует ответу на запрос «Получение информации об объекте слоя», кроме состава полей в свойстве "fields", т. к. текущий запрос может не иметь данного свойства (не указан параметр "fields" в запросе), либо иметь

ограниченный набор полей (в параметре "fields" перечислены не все поля структуры слоя).

3.1.7.3 Получение информации об объекте слоя

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/<object_id>/

Необязательные параметры запроса:

− returnBbox – условие включения в ответ габаритов объекта. "0» – не включать (значение по умолчанию); "1" – включить bbox в ответ;

− bboxSR – spatial reference для габаритов объекта в формате EPSG. Возможные значения: 4326 (значение по умолчанию) и 3857;

− returnGeom – условие включения в ответ геопространственных данных объекта. Геопространственные данные в ответе будут возвращены в формате GeoJSON. Возможные значения: "0" – не включать (значение по умолчанию); "1" – включить геопространственные данные в ответ;

− geomSR – spatial reference для геопространственных данных объекта в формате EPSG. Возможные значения: 4326 (значение по умолчанию) и 3857;

− geomZoom – параметр, учитываемый при запросе геопространственных данных объекта. При указании текущего зума, геопространственные данные будут автоматически упрощены под указанный зум. Если параметр не задавать, то геопространственные данные будут в исходном виде – без упрощения;

− lng – код языка, для которого будут получены локализованные значения в полях свойства "fields". Если параметр не указан, то будут получены значения для языка по умолчанию.

Ответ:

{
"id": 1,

```
    "fields": {
      "field_1": "Значение колонки 1",
      "field_2": "Значение колонки 2",
      "field_3": 3
    },
    "geom": {
      "type": "Polygon",
      "coordinates": [
        [
          [ 39.674470691019003, 45.201854294024443 ],
          [ 39.672229085150633, 45.164807545385699 ],
          [ 39.728412719962421, 45.151394461232584 ],
          [ 39.674470691019003, 45.201854294024443 ]
        ]
      ]
    },
    "bbox": {
      "sw" : [39.67222, 45.20185],
      "ne" : [39.72841, 45.15139]
    }
  }
```


где:

− id – уникальный идентификатор объекта. Соответствует колонке "gis_id". Всегда присутствует в ответе;

− geom – геопространственные данные объекта в формате GeoJSON. Может отсутствовать в ответе: в зависимости от значения параметра "returnGeom". Проекция зависит от параметра "geomSR";

− bbox – габариты объекта. Может отсутствовать в ответе: в зависимости от значения параметра "returnBbox". Проекция зависит от параметра "bboxSR";

− fields – свойство, значением которого является объект формата JSON. Каждое свойство объекта описывает одно поле (колонку) объекта. Ключ каждого поля – это код колонки слоя, а тип значения зависит от типа колонки.

Если запрошен несуществующий объект, то ответ будет:

```
{
   "id": 1000,
   "fields": {}
}
```

где:

− id – запрошенный несуществующий идентификатор объекта;

− fields – всегда пустой объект формата JSON.

3.1.7.4 Создание объекта слоя

Запрос:

POST http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/

Для выполнения запроса необходим токен авторизации. Параметры создаваемого объекта передаются в формате JSON в теле запроса:

```
{
   "fields": {
      "field_1": "Значение колонки 1",
      "field_2": "Значение колонки 2",
      "field_3": 3
   },
   "geom": {
      "type": "Polygon",
      "coordinates": [
         [
            [ 39.674470691019003, 45.201854294024443 ],
            [ 39.672229085150633, 45.164807545385699 ],
```

[39.728412719962421, 45.151394461232584],
[39.674470691019003, 45.201854294024443]
]
]
}
}

где:

− fields (обязательный) – свойство, значением которого является объект JSON. Если у объекта нет обязательных полей, то может быть пустым. Если у слоя, в котором хранится объект, есть колонки обязательные для заполнения (required = true), то они должны быть перечислены;

− geom (необязательный) – геопространственные данные объекта в формате GeoJSON. Если не указан, то создаётся объект без геопространственных данных;

− lng – код языка, для которого будут изменены локализованные значения в полях объекта. Если параметр не указан, то будут изменены значения для языка по умолчанию.

Если слой имеет локализованную по значению колонку и в неё необходимо установить значение для определенного языка, то в свойстве "fields" указывается код колонки, а язык передается параметром запроса "lng", например, "lng=ru". Если в "fields" указан код локализованной колонки и не указан параметр "lng", то значение будет установлено для языка по умолчанию.

Ответ:

Ответ на запрос создания объекта слоя по структуре соответствует ответу на запрос «Получение информации по объекту слоя».

3.1.7.5 Изменение объекта слоя

Запрос:

PUT http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/<object_id>/

Для выполнения запроса необходим токен авторизации.

Формат запроса на изменение объекта соответствует формату запроса на создание объекта слоя, с отличием – обязательные для заполнения поля не указываются, если они не будут изменяться.

Ответ:

Ответ на запрос изменения объекта слоя по структуре соответствует ответу на запрос «Получение информации по объекту слоя». Если запрошен несуществующий объект, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.7.6 Удаление объекта слоя

Запрос:

DELETE http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/<object_id>/

Для выполнения запроса необходим токен авторизации.

Ответ: true.

Если запрошен несуществующий объект, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.7.7 Получение геопространственных данных объекта слоя

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/<object_id>/geom/

Для выполнения запроса необходим токен авторизации.

Ответ:

{
"type": "Point",

"coordinates": [3673543.19617802882567, 3895303.963393894489855]

}

В ответом показаны геопространственные данные объекта в формате GeoJSON.

Если запрошены геопространственные данные несуществующего объекта, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.7.8 Изменение геопространственных данных объекта слоя

Запрос:

PUT http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/<object_id>/geom/

Для выполнения запроса необходим токен авторизации. Геопространственные данные в GeoJSON-формате передаются в теле запроса.

Необязательные параметры запроса:

− geomSR – spatial reference для геопространственных данных объекта в формате EPSG. Возможные значения: 4326 (значение по умолчанию) и 3857

{

"coordinates": [61.33891, 35.83611],

"type": "Point"

}

Ответ:

Ответ на запрос изменения геопространственных данных объекта слоя по структуре соответствует ответу на запрос «Получение геопространственных данных объекта».

Если запрошено изменение геопространственных данных несуществующего объекта, то сервер вернет ошибку со статусом 404. Описание ошибок приведено в п. 3.1.12.

3.1.8 Работа с файлами

Файлы можно добавлять только к объектам, имеющим файловые поля. Код поля, в которое необходимо добавить файл, указывается одним из параметров при запросе. Если необходимо добавить файл в локализованную по значению колонку, то указывается параметр запроса языка "lng", например, "lng=ru". Если параметр "lng" не указывать при добавлении файла в локализованную по значению колонку, то файл будет добавлен в значение языка по умолчанию.

3.1.8.1 Добавление файла к объекту

Запрос:

POST http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/<object_id>/files/

Для выполнения запроса необходим токен авторизации. Обязательные параметры запроса:

- field – код колонки, в которую необходимо добавить файл;
- file – файл в параметре POST-запроса.

Ответ:

{
"name": "имя загруженного файла.m4a",
"path": "//<host>/static/content_files/gis_db/0bb97e774aee56cf3e7b0b0717ab62ce.m4a",
"mime": "audio/x-m4a",
"id": 3,
"uuid": "754279f7-fdf1-4f75-a5d8-1bbd18c3dd5b"

}

где:

- name – имя загруженного файла;
- path – путь, по которому можно запросить файл;
- mime – mime-тип файла;
- id – уникальный идентификатор файла;
- uuid – uuid файла.

3.1.8.2 Удаление файла объекта

Код поля, из которого удаляется файл, указывается одним из параметров при запросе. Если необходимо удалить файл из локализованной по значению колонки, то указывается язык параметром запроса "lng", например, "lng=ru2. Если параметр "lng" не указать при удалении файла из локализованной по значению колонки, то файл будет удален из значения языка по умолчанию.

Запрос:

DELETE http://<host>/api/2.7/<project_code>/<map_code>/layers/<layer_code>/objects/<object_id>/files/<file_id>/

Для выполнения запроса необходим токен авторизации. Обязательные параметры запроса:

- field – Код колонки, из которой необходимо удалить файл.

Ответ: null.

3.1.9 Поиск по слоям и метаданным

Запрос:

GET http://<host>/api/2.7/<project_code>/<map_code>/search/

Запрос выполняется по полному или частичному совпадению слов во фразе и данных для поиска.

При поиске по слоям учитывается поле "name". Поиск по метаданным (объектам слоев) использует настройки, которые можно сохранить при публикации карты. Результаты отсортированы по приоритетам, затем по типу, затем по значению. Если найдено достаточное количество объектов с точным совпадением, то поиск не продолжается (другие слои, ниже по списку в настройках поиска, не участвуют). При равных приоритетах слои важнее объектов. Параметры:

- "30" – точное совпадение фразы и данных объекта;
- "20" – все слова из фразы присутствуют в данных;
- "10" – присутствуют некоторые слова из фразы.

Обязательные параметры запроса:

- phrase – фраза поискового запроса.

Необязательные параметры запроса:

- lang – код языка для поиска по локализованным названиям слоев. По умолчанию – "ru";
- limit – количество элементов в результатах поиска. По умолчанию – "3".

Ответ:


```
[
   {
      "id": 103,
      "code": "visual",
      "name": "Визуализация для поиска",
      "priority": 20,
      "type": "layer",
      "icon": "/static/geoicons/visual.png"
   },
   {
      "id": 101,
      "code": "layer",
```

```
    "name": "Слой для поиска",
    "priority": 20,
    "type": "layer",
    "icon": "/static/geoicons/polygon.png"
  },
  {
    "name": 3,
    "gis_id": 3,
    "priority": 20,
    "layer": "layer",
    "type": "object",
    "name_column": "gis_id",
    "search_column": "data",
    "layer_name": "Слой для поиска",
    "layer_icon": "/static/geoicons/polygon.png",
    "value": ""
  },
  {
    "name": 8,
    "gis_id": 8,
    "priority": 20,
    "layer": "layer",
    "type": "object",
    "name_column": "gis_id",
    "search_column": "data",
    "layer_name": "Слой для поиска",
    "layer_icon": "/static/geoicons/polygon.png",
    "value": ""
  }
]
```

где:

- type – тип элемента результата;
    1) layer – слой карты;
    2) object – объект слоя.

Поля слоя карты:

- id – идентификатор слоя в системе;
- code – строковый код слоя;
- icon – путь до иконки слоя;
- type – тип элемента результата;
- name – значение имени слоя, которое участвовало в поиске;
- priority – значение приоритета при поиске.

Поля объекта слоя:

- gis_id – идентификатор объекта в слое;
- priority – значение приоритета при поиске;
- type – тип элемента результата;
- layer – строковый код слоя;
- layer_name – имя слоя;
- layer_icon – путь до иконки слоя;
- name – значение имени объекта;
- name_column – строковый код поля названия объекта;
- search_column – строковый код поля объекта для поиска;
- value – значение поля объекта, которое участвовало в поиске.

3.1.10 Запрос изображений

Позволяет изменить размер любого изображения, привязанного к любому объекту в системе. Например, создать уменьшенную копию изображения.

Запрос:

GET http://<host>/api/2.7/<project_code>/image/<file_uuid>/

Необязательные параметры запроса:

− w – желаемая ширина итогового изображения; если "h" не будет указан, изображение изменится с сохранением пропорций;

− h – желаемая высота итогового изображения; если "w" не будет указан, изображение изменится с сохранением пропорций;

− fit – параметр преобразования изображения (работает аналогично CSS3 object-fit). Применяется, если указаны оба параметра: "w" и "h". Допустимые значения: "none", "fill", "contain", "cover", "scale-down". По умолчанию – "fill";

− f – желаемый формат изображения. Допустимые значения: "png", "jpg", "jpeg". По умолчанию используется исходный формат изображения.

3.1.11 Запрос тайлов

Запрос:

GET http://<host>/tms/<project_code>/<map_code>/?x={x}&y={-y}&z={z}

Дополнительные необязательные параметры запроса:

− base – условие включения в тайл базовую карту: включать – "1" или не включать – "0" (по умолчанию – "1");

− layers – список идентификаторов слоёв через запятую;

− retina – множитель Retina-экранов (retina=2 выдаст тайлы 512x512);

− lng – язык текстов на карте;

− q – качество тайлов, "0" – полноцветное изображение PNG, "1" – ограниченная палитра цветов PNG8 (по умолчанию – "0").

3.1.12 Обработка ошибок

HTTP-статусы выполнения запросов:

− HTTP 200 – запрос выполнен успешно; в ответе json-объект с результатом операции;

− HTTP 400 – некорректный запрос;

− HTTP 401 – ошибка аутентификации: выполнение данного запроса требует аутентификации;

− HTTP 403 – ошибка авторизации: выполнение данной операции для данного пользователя запрещено правами доступа;

− HTTP 404 – не найдена сущность, соответствующая запросу;

− HTTP 405 – недопустимый метод HTTP;

− HTTP 500 – в результате выполнения запроса произошла ошибка; в ответе json-объект описания ошибки:

```
{
   "error": "error|warning|notice",
   "code": "код_ошибки",
   "message": "Описание ошибки выполнения запроса",
   "data": {}
}
```

где:

data – всегда json-объект, который может содержать дополнительную информацию по ошибке. Например, список обязательных параметров запроса или колонок, установление значения для которых вызывало ошибку при редактировании объекта.

Ответы со статусами "400", "401", "403", "404", "405" приходят в виде HTML.

Ответы со статусами "200" и "500" приходят в виде объектов формата JSON.

3.2 Настройка работы программы через JS API

Базовый шаблон HTML для подключения JS API.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>OMJS</title>
```

```
        <link rel="stylesheet"
href="http://example.com/static/omjs/2.8/omjs.css" />
        <script src="http://example.com/static/omjs/2.8/omjs.js"></script>
         <script>
        // Your code here
         </script>
    </head>
    <body>
        <div style="width:600px; height:400px;" id="map"></div>
    </body>
    </html>
```

3.2.1 Класс Project

Класс Программы, реализующий работу с данными опубликованной карты.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
omjs.initProject (<String|Project config>) async	omjs.Project	Инициализация проекта с конфигурацией. Можно вместо объекта передать URL к JSON с конфигурацией
new **omjs.Project**(<Project config>)	omjs.Project	Инициализация проекта с объектом конфигурации

Конфигурация:

Опция	Тип	Описание
apiUrl req	String	Url к Ситроникс.ГИС REST API
mapCode req	String	Код опубликованной карты Ситроникс.ГИС
token	String	Токен авторизации
lng	String	Код языка карты. Если не передан, Ситроникс.ГИС вернет данные на основном языке карты

Методы:

Метод	Возвращает	Описание
login(<String> *user*, <String> *password*) async	ajax response	Авторизирует пользователя на проекте
logout() async	ajax response	Завершает сессию пользователя
loadContainers(<loadContainers Options>) async	omjs.ContainersCollection | ajax response	Загружает список контейнеров
loadLayers() async	omjs.ContainersCollection	Загружает полный список контейнеров, но возвращает коллекцию, содержащую только слои
loadLayer(<String> *код слоя*) async	omjs.Project.Layer	Загружает слой по его коду
loadObjects(<String> *код слоя*, <loadObjects Options>) async	omjs.Collection | ajax response	Загружает список объектов слоя. Возвращает коллекцию объектов omjs.Project.LayerObject
loadObject(<String> *код слоя*, <Number> *id объекта*, <loadObject Options>) async	omjs.Project.LayerObject | ajax response	Загружает объект слоя

loadContainers Options:

Опция	Тип	По умолчанию	Описание
raw	Boolean	false	Вернуть "сырой" ответ сервера или предварительно обработать и вернуть коллекцию контейнеров
type	String	'full'	Тип запрашиваемого списка контейнеров: 'full' – Полный список контейнеров опубликованной карты; 'publish' – Публичный список контейнеров опубликованной карты

loadObjects Options:

Опция	Тип	По умолчанию	Описание
raw	Boolean	false	Вернуть "сырой" ответ сервера или предварительно обработать и вернуть коллекцию объектов
fields	String[]	[]	Список кодов колонок, значения которых также попадут в результат. Если необходимо получить все колонки, то нужно указать первым элементом массива '*'
fcode	String	null	Код колонки для фильтрации
fval	String	null	Значение фильтруемой колонки
limit	Number	null	Ограничение на количество выбранных объектов
sort	String	null	Набор колонок, по которым будет выполнена сортировка списка. Значение представляется в формате name,a|capital|id,d, где a=ASC (по возрастанию), d=DESC (по убыванию). Направление указывать не обязательно, по умолчанию ASC
returnBbox	Number	0	Условие включения – ответ Bounding Box объекта – принимает значения "1" или "0"
bboxSR	Number	4326	Spatial reference для Bounding Box объекта в формате EPSG. Принимает значения 4326 или 3857
returnGeom	Number	0	Загружать ли геометрию объектов (если 1, геометрия будет возвращена в ответе в формате GeoJson)

geomSR	Number	4326	Spatial reference для геометрии объекта в формате EPSG. Принимает значения 4326 или 3857
zoom	Number	null	Параметр, учитываемый при запросе полной геометрии объекта: в случае указания текущего зума геометрия будет автоматически упрощена под указанный зум; если параметр не задавать, геометрия будет в исходном виде (без упрощения)

3.2.2 Класс Container

Базовый класс, представляющий контейнер Ситроникс.ГИС. Унаследован от "omjs.Model".

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Project.Container**(<Object> attributes, <Container Options>)	omjs.Project.Container	Создаёт экземпляр контейнера. Принимает объект с атрибутами контейнера (метаданные) и экземпляр проекта

Опции:

Опция	Тип	Описание
project req	omjs.Project	Экземпляр проекта опубликованной карты Ситроникс.ГИС. Используется дочерними классами для доступа к данным

Свойства:

Свойство	Тип	Описание
project	omjs.Project	Экземпляр проекта опубликованной карты Ситроникс.ГИС

3.2.3 Класс ContainersCollection

Коллекция контейнеров "omjs.Project.Container". Унаследован от "Collection".

Методы:

Метод	Возвращает	Описание
loadObjects(<loadObjects Options>) async	omjs.Collection	Загружает объекты всех слоёв в коллекции. Возвращает коллекцию объектов omjs.Project.LayerObject

3.2.4 Класс Layer (Data)

Класс, представляющий слой Ситроникс.ГИС. Унаследован от "omjs.Project.Container".

Методы:

Метод	Возвращает	Описание
loadObjects(<loadObjects Options>) async	omjs.Collection	Загружает объекты слоя. Возвращает коллекцию объектов omjs.Project.LayerObject
loadObject(<Number> *id объекта*, <loadObject Options>) async	omjs.Project.LayerObject	Загружает объект слоя

Свойства:

Свойство	Тип	Описание
objects	omjs.Collection	Коллекция объектов omjs.Project.LayerObject с момента последней загрузки

3.2.5 Класс LayerObject

Класс, представляющий объект слоя Ситроникс.ГИС. Унаследован от "omjs.Model". Хранит в себе все данные объекта, включая геометрию (если она есть).

Класс, представляющий объект слоя платформы Ситроникс.ГИС. Унаследован от "omjs.Model". Хранит в себе все данные объекта, включая геометрию.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Project.LayerObject**(<Number> *id*, <Object> *fields?*, <LayerObject options?>)	omjs.LayerObject	Инициализация объекта с заданным id и опциональным набором колонок

Опции:

Опция	Тип	Описание
bbox	Object	Bounding Box объекта в формате: { "sw":[39.72841, 45.20185], "ne":[39.67222, 45.15139] }
geometry	Object	Объект геометрии в формате GeoJSON

Методы:

Метод	Возвра-щает	Описание
getId()	Number	Возвращает id объекта
getGeometry()	Object	Возвращает объект геометрии в формате GeoJSON
setGeometry(<Object> *Geometry*)	this	Присваивает геометрию объекту. Принимает объект геометрии в формате GeoJSON
getBBox()	Object	Возвращает Bounding Box объекта
setBBox(<Array> *sw*, <Array> *ne*)	Object	Устанавливает Bounding Box объекта. Принимает два массива с координатами юго-западной и северо-восточной точки BBox – например, ".setBBox([39.72841, 45.20185], [39.67222, 45.15139])"

3.2.6 Класс View

Класс для отображения карт платформы Ситроникс.ГИС. Является прослойкой между данными Ситроникс.ГИС и библиотекой OpenLayers.

Использует "omjs.Project" для доступа к данным опубликованной карты Ситроникс.ГИС. Библиотека для отображения карт – OpenLayers.

Инициализация:

Фабрика/Конструктор	Возвра-щает	Описание
omjs.initView(<HTMLElement|String> *CSS Селектор*, <String|View config>, <View config?>) async	omjs.View	Инициализация представления карты в заданном элементе. Можно вместо объекта передать URL к JSON с конфигурацией, в этом случае дополнительно можно передать ещё один объект с приоритетными опциями. В конфигурации нужно также указать опции проекта (класс Project)
***new* omjs.View**(<HTMLElement|String> *CSS Селектор*, <omjs.Project>, <View config?>)	omjs.View	Инициализация представления в заданном элементе. Принимает на вход HTML элемент (или CSS селектор элемента), в котором нужно отобразить карту, экземпляр проекта (класс Project) и объект с настройками (Options) View

Опции:

Опция	Тип	По умолчанию	Описание
tilesUrl req	String	' '	Шаблон URL, по которому omjs сможет запрашивать тайлы Ситроникс.ГИС. Например: "http://example.com/tms/<project_code>/{map_code}/?layers={layers}&x={x}&y={-y}&z={z}"
show	Boolean	true	Условие отображения карты при инициализации
zoom	Number	2	Начальный zoom карты
minZoom	Number	1	Минимальный zoom карты
maxZoom	Number	22	Максимальный zoom карты
center	[]	[37.6155600, 55.7522200]	Широта и долгота центра карты
publication.baseMapType	Number	omjs.baseLayerType.MAP	Тип подложки карты, отображаемой по умолчанию: omjs.baseLayerType.MAP – опубликованная карта Ситроникс.ГИС; omjs.baseLayerType.URL – шаблон url-адреса тайлов; omjs.baseLayerType.EXTERNAL – внешний сервис (Google, Yandex, ...)
publication.base	String	" "	Значение подложки карты, отображаемой по

MapValue			умолчанию. Для типа подложки omjs.baseLayerType.MAP – код опубликованной карты, используемой в качестве подложки. Для типа подложки omjs.baseLayerType.URL – url-адрес шаблона тайлов. Для типа подложки omjs.baseLayerType.EXTERNAL возможные значения: 'OSM', 'kosmosnimki', 'yandexMap', 'yandexSat', 'yandexHibrid', 'yandexPublic', 'googleRoadmap', 'googleSat', 'googleHibrid', 'googleTerrain'
baseLayer	String	'gis'	Подложка, которая будет использована при начальном отображении карты. По умолчанию – настроенная подложка карты Ситроникс.ГИС. Поддерживаемые подложки: 'gis' – Ситроникс.ГИС; 'kosmosnimki' – Космоснимки; 'yandex' – Яндекс.Схема; 'yandexSat' – Яндекс.Спутник; 'google' – Google Карта
lng	String	'ru'	Язык интерфейса
translations	Object	{}	Переопределение предустановленных фраз текущего языка интерфейса. {'Объекты': 'Map objects', 'Адрес': 'Location'}

Методы:

Метод	Возвращает	Описание
getContainer()	jQuery Element	Возвращает контейнер (HTML элемент), в котором отображена карта
getMap()	ol.Map	Возвращает экземпляр карты "OpenLayers", который используется для отображения карты
getProject()	omjs.Project	Возвращает экземпляр проекта (класс Project), который используется для доступа к данным Ситроникс.ГИС в текущей карте
getLayers()	omjs.View.Layer Collection	Возвращает коллекцию слоёв "omjs.View.Layer" карты
getWidgetManager()	WidgetManager	Возвращает менеджер виджетов карты

show()	this	Отображает карту, если она скрыта
hide()	this	Скрывает карту
showGisPublication(<Function>beforeVectorAdd) async		Отображает опубликованную карту в соответствии с настройками публикации. Функция beforeVectorAdd будет вызвана перед каждым добавлением векторного слоя для предоставления возможности изменения конфигурации слоя
addLayer(<String> *Код слоя*, <Object> *Опции слоя?*) async	omjs.View.Layer	Загружает данные слоя из Ситроникс.ГИС. Создаёт слой "omjs.View.Layer" и добавляет его на карту. Можно передать несколько кодов слоёв Ситроникс.ГИС через пробел (при установленном параметре layerType='tile'). В опциях слоя можно указать тип слоя "layerType": 'tile' – создаст тайловый слой "omjs.View.TileLayer" (по умолчанию); 'vector' – создаст векторный слой "omjs.View.VectorLayer". Поддерживаются следующие типы слоёв для визуализации данных: "gis_vector", "gis_vector_category", "gis_chart_layer", "gis_vector_feature", "gis_mvt"
addLayer(<omjs.View.Layer> *Слой*) async	omjs.View.Layer	Добавляет слой на карту
addLayer(<ol.layer.Layer> *Слой*) async	omjs.View.Layer	Создаёт слой "omjs.View.Layer" на основании переданного слоя OpenLayers и добавляет его на карту
removeLayer(<omjs.View.Layer> *Слой*)	this	Удаляет слой с карты
getBaseLayer()	BaseLayer	Возвращает менеджер подложек
switchBaseLayer(<Number> *ID подложки*)	null	Переключает базовую подложку. Принимает ID базовой подложки из конфигурации публичной карты
showBalloon(<Balloon Options> *Опции?*)	ol.Overlay	Отображает балун (balloon) на карте. Если в момент открытия балуна на карте уже есть открытый балун, то

		старый балун закрывается и открывается новый. Возвращает "ol.Overlay"
hideBalloon()	null	Скрывает балун
addMarker(<Array> *Координаты*, <addMarker Options> *Опции?*)	ol.Feature	Отображает маркер на карте. Возвращает "ol.Feature"
getMarkerGroup(<String> *Имя группы?*)	omjs.View.Marker Layer	Возвращает группу маркеров по имени. Если не передать имя группы, вернёт группу 'default'

Balloon Options: Любые опции, поддерживаемые "ol.Overlay"

Опция	Тип	По умолчанию	Описание
content	String | DOMElement	' '	Содержимое балуна
class	String	' '	Дополнительные CSS классы, перечисленные через пробел, которые будут добавлены к контейнеру балуна
autoPan	Boolean	true	Если установлено в "true", в момент отображения балуна карта сдвинется так, чтобы балун полностью поместился в область видимости
position	Array	undefined	Координата в проекции 'EPSG:4326', в которой нужно отобразить балун
positioning	String	'bottom-center'	Определяет, как элемент балуна будет расположен относительно позиции балуна на карте. Возможные значения: 'bottom-left', 'bottom-center', 'bottom-right', 'center-left', 'center-center', 'center-right', 'top-left', 'top-center', 'top-right'

addMarker Options:

Опция	Тип	По умолчанию	Описание
icon	String	' '	URL изображения иконки
group	String	'default'	Имя группы маркеров. Маркеры в группе собираются в кластеры

3.2.7 Класс Layer (View)

Используется для отображения слоя на карте. Унаследован от "omjs.Model". Может использовать "omjs.Project.Layer" для работы с данными слоя Ситроникс.ГИС.

Также использует "ol.layer.Layer" и "ol.source.Source" для отображения слоя на карте.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.View.Layer**(<View Layer options>)	omjs.View.Layer	Создаёт экземпляр визуального слоя

Опции:

Опция	Тип	Описание
attributes	Object	Объект с атрибутами. Может быть любым набором данных
projectLayers	omjs.Project.ContainersCollection	Коллекция слоёв Ситроникс.ГИС. Используется дочерними классами

Методы:

Метод	Возвращает	Описание
show()	undefined	Отображает слой на карте
hide()	undefined	Скрывает слой с карты
toggle()	undefined	Отображает или скрывает слой, в зависимости от того, отображен слой или нет

isVisible()	Boolean	Отображается слой на карте или нет
getProjectLayers()	omjs.Project.Containers Collection	Возвращает экземпляр коллекции слоёв проекта
getMapLayer()	ol.layer	Возвращает экземпляр слоя "OpenLayers", с которым ассоциирован слой "omjs.View.Layer"
setMapLayer(<ol.layer.Layer>)	undefined	Устанавливает слой OpenLayers
setSource(<ol.source.Source>)	undefined	Устанавливает источник данных "OpenLayers"

3.2.8 Класс LayerCollection

Коллекция визуальных слоёв. Унаследован от "omjs.Collection".

3.2.9 Класс TileLayer

Используется для отображения тайлового слоя на карте. Унаследован от "omjs.View.Layer".

Может использовать сразу несколько слоёв "omjs.Project.Layer" для отображения нескольких слоёв одним тайловым слоем. Также использует "ol.layer.Tile" и "ol.source.Tile" для отображения тайлового слоя на карте.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.View.TileLayer**(<View Layer options>)	omjs.View.TileLayer	Создаёт экземпляр визуального тайлового слоя

Методы:

Метод	Возвращает	Описание
setActiveLayers(<String> layer codes)	null	Устанавливает набор слоёв Ситроникс.ГИС для тайлового слоя карты. Принимает строку с перечисленными через пробел кодами слоёв

3.2.10 Класс VectorLayer

Используется для отображения векторного слоя на карте. Унаследован от "omjs.View.Layer".

Может использовать "omjs.Project.Layer" для отображения слоя Ситроникс.ГИС на карте. Также использует "ol.layer.Vector" и "ol.source.Vector" для отображения векторного слоя на карте.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.View.VectorLayer(** <View Layer options>)	omjs.View.VectorLayer	Создаёт экземпляр визуального векторного слоя

Опции:

Опция	Тип	По умолчанию	Описание
clustering	Number	null	При назначении параметра точечный слой будет кластеризован и "источник данных" будет изменён на "ol.source.Cluster" Указывается в виде максимальной дистанции в пикселях между кластерами
projectLayers	omjs.Project .Containers Collection	null	Коллекция слоёв Ситроникс.ГИС. Первый слой этой коллекции будет использован как источник геометрии

Методы:

Метод	Возвращает	Описание
setStyle(<ol.style.Style> | <ol.style.Style[]> | <ol.FeatureStyleFunction>)	null	Устанавливает стили векторного слоя. В случае с "ol.FeatureStyleFunction", функция должна вернуть массив стилей "OpenLayers"

События:

Событие	Данные	Описание
click	'feature' – ol.Feature 'olEvent' – Объект события OpenLayers	Срабатывает, когда пользователь кликает по

		геообъекту
singleclick	'feature' – ol.Feature 'olEvent' – Объект события OpenLayers	Срабатывает через 250 мс после того, как пользователь кликнул по геообъекту
dblclick	'feature' – ol.Feature 'olEvent' – Объект события OpenLayers	Срабатывает, когда пользователь дважды кликает по геообъекту
pointermove	'feature' – ol.Feature 'olEvent' – Объект события OpenLayers	Срабатывает, когда пользователь двигает курсор по геообъекту
pointerenter	'feature' – ol.Feature 'olEvent' – Объект события OpenLayers	Срабатывает, когда пользователь навёл курсор на геообъект
pointerleave	'feature' – ol.Feature 'olEvent' – Объект события OpenLayers	Срабатывает, когда пользователь увёл курсор с геообъекта

3.2.11 Класс Choropleth

Используется для отображения картограмм на карте. Унаследован от "omjs.View.VectorLayer".

Использует "omjs.Project.Layer" для доступа к геометрии слоя Ситроникс.ГИС. Также использует "ol.layer.Vector" и "ol.source.Vector" для отображения векторного слоя на карте.

По умолчанию использует данные самого слоя Ситроникс.ГИС и не классифицирует данные для отображения картограммы. Есть возможность классифицировать данные для более тонкой настройки картограммы.

Инициализация:

Фабрика/Конструктор	**Возвращает**	**Описание**
new **omjs.View.Choropleth(** <omjs.Project.Layer> *слой Ситроникс.ГИС Server*, <Choropleth options>)	omjs.View.Choropleth	Создаёт экземпляр слоя – картограммы. Объекты слоя Ситроникс.ГИС должны быть полностью загружены вместе с геометрией. Опции нужны только для отображения картограмм с неклассифицированными данными

Опции:

Опция	Тип	По умолчанию	Описание
minFillColor	String	'#ffffff'	Цвет для минимального значения набора неклассифицированных данных картограммы
maxFillColor	String	'#000000'	Цвет для максимального значения набора неклассифицированных данных картограммы
totalColors	Number	10	Количество цветов в градиенте от minFillColor до maxFillColor
strokeColor	String	'#ffffff'	Цвет контура объектов
strokeWidth	Number	1	Толщина контура объектов

Методы:

Метод	Возвра-щает	Описание
setField(<String> *код колонки*)	this	Берёт код слоя Ситроникс.ГИС, генерирует и отображает картограмму с неклассифицированными данными по переданному коду колонки
bindData(<Any> *данные*, <Function(data, obj, feature)> *функция сопоставления*)	this	Устанавливает новый набор данных для картограммы. Набор данных необходимо сопоставить с данными векторного слоя. Для этого вторым параметром нужно передать функцию сопоставления данных, которая принимает следующие значения: data – данные, которые установлены первым параметром; obj – объект слоя Ситроникс.ГИС, которые используется картограммой; feature – "ol.Feature". Функция должна вернуть значение из "data"

classify(<Object[]> *массив классификаторов*)	this	Функция для классификации данных картограммы. Принимает массив объектов – классификаторов. Классификатор — это объект следующего формата:
		{
		// Название классификатора
		name: '10-20',
		// Стили классификатора OpenLayers
		style: new omjs.ol.style.Style({
		fill: new omjs.ol.style.Fill({
		color: '#fff000'
		})
		}),
		// Функция, которая определяет, попадает значение
		// в данный классификатор или нет.
		// Должна вернуть true, если попадает,

		// или false, если не попадает.
		matcher: function(value){
		return value > 10 && value <= 20;
		}
		}
		Каждый объект векторного слоя картограммы будет причислен к одному из классификаторов, после чего к этому объекту будут применены стили классификатора. Стили классификаторов применяются после установки классификаторов

getLegendData()	Array	Возвращает список объектов содержимого легенды "omjs.Widget.Legend". Применяется только для неклассифицированной картограмма

3.2.12 Класс MarkerLayer

Используется для отображения группы маркеров на карте. Унаследован от "omjs.View.VectorLayer".

Также использует "ol.layer.Vector" и "ol.source.Vector" для отображения векторного слоя на карте.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.View.MarkerLayer**(<Marker Layer options>)	omjs.View.MarkerLayer	Создаёт экземпляр слоя группы маркеров

Опции:

Опция	Тип	По умолчанию	Описание
group	String	'default'	Имя группы маркеров. Маркеры в группе собираются в кластеры
clustering	Number	null	При указании параметра точечный слой будет кластеризован и "источник данных" будет изменён на "ol.source.Cluster". Указывается в виде максимальной дистанции в пикселях между кластерами

Методы:

Метод	Возвращает	Описание
addMarker(<Array> *Координаты* | ol.Feature, <addMarker Options> *Опции?*)	ol.Feature	Добавляет маркер в группу маркеров
getMarkers()	Array ol.Feature	Возвращает массив маркеров
setClusterStyle(<Object | ol.Style> *Стили*)	this	Устанавливает стили для кластеров. Может принимать объект с параметром `icon` – URL изображения иконки. Либо может принять объект стилей OpenLayers "ol.Style"

3.2.13 Класс BaseLayer

Менеджер подложек. Используется для переключения различных подложек на карте, таких как Ситроникс.ГИС, Yandex, Google и др.

Экземпляр класса создаётся при инициализации "omjs.View" и доступен через метод "view.getBaseLayer()".

Методы:

Метод	Возвращает	Описание
show()	null	Отображает подложку на карте
hide()	null	Скрывает подложку карты
switchTo(<String> *название подложки*)	null	Переключает подложку. Поддерживаемые подложки: 'BaseSource' – Базовая карта; 'OSM' – OpenStreetMap; 'kosmosnimki' – Космоснимки; 'yandexMap' – Яндекс Схема; 'yandexSat' – Яндекс Спутник; 'yandexHibrid' – Яндекс Гибрид; 'yandexPublic' – Яндекс народная карта; 'googleRoadmap' – Google Карта; 'googleSat' – Google Спутник; 'googleHibrid' – Google Гибрид; 'googleTerrain' – Google Рельеф

3.2.14 Класс WidgetManager

Менеджер виджетов является коллекцией классов-виджетов и предназначен для добавления виджетов на карту. Все виджеты добавляются с помощью этого класса.

Экземпляр класса создаётся при инициализации "omjs.View" и доступен через метод "view.getWidgetManager()".

Методы:

Метод	Возвращает	Описание
addWidget(<String|omjs.Widget> *widget*, <Object> *options?*)	this	Добавляет виджет на карту. Может принимать код предустановленного виджета в виде строки: 'Button', 'ButtonGroup', 'ZoomControl', 'LayerSwitcher', 'DistanceButton', 'AreaButton' Является объектом опций для инициализации виджета. Также может принять экземпляр класса виджета "omjs.Widget"
hideWidgets()	null	Скрывает все виджеты
showWidgets()	null	Отображает все виджеты

3.2.15 Класс Widget

Базовый класс виджета. От этого класса наследуются все виджеты "omjs".

Содержит в себе необходимую минимальную логику для создания виджета.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Widget**(<Object> *options*?)	omjs.Widget	Создаёт новый экземпляр виджета. Создаёт HTML Элемент – контейнер, который можно будет использовать для рендера виджета. В объект опций можно передать 'classes' – список CSS классов, перечисленные через запятую, которые будут добавлены к контейнеру

Опции:

Опция	Тип	По умолчанию	Описание
classes	String	''	Список CSS классов, перечисленные через запятую, которые будут добавлены к контейнеру
css	Object	null	Объект стилей CSS в формате jQuery, которые будут применены к контейнеру виджета

Методы:

Метод	Возвращает	Описание
getContainer() jQuery	HTMLElement	Возвращает DOM Элемент – контейнер виджета
show()	this	Отображает виджет
hide()	this	Скрывает виджет
onAdd(<omjs.View>)	null	Функция-заглушка для дочерних классов. Вызывается при добавлении виджеат в "WidgetManager". В функцию будет передан экземпляр "omjs.View"

3.2.16 Класс Button

Используется для добавления различных кнопок на карту. Унаследован от "omjs.Widget".

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Widget.Button**(<Object> *widget options*?)	omjs.Widget.Button	Создаёт новый экземпляр виджета-кнопки

Методы:

Метод	Возвращает	Описание
on(<String> *eventType*, <Function> *handler*)	this	Регистрирует обработчик handler DOM события "eventType" на контейнер виджета

3.2.17 Класс ButtonGroup

Используется для визуальной группировки виджетов-кнопок. Унаследован от "omjs.Widget".

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Widget.ButtonGroup**(<Object> *options*?)	omjs.Widget.ButtonGroup	Создаёт новый экземпляр группы кнопок

Опции:

Опция	Тип	По умолчанию	Описание
orientation	String	'horizontal'	Ориентация группы кнопок. Вертикальная 'vertical' или горизонтальная 'horizontal'
buttons	omjs.Widget.Button[]	null	Массив виджетов-кнопок, которые будут добавлены в группу, как только она будет создана

Методы:

Метод	Возвращает	Описание
add(<omjs.Widget.Button)	this	Добавляет кнопку в группу
remove(<omjs.Widget.Button)	this	Удаляет кнопку из группы

3.2.18 Класс DistanceButton

Встроенный виджет, предоставляющий инструмент измерения расстояния на карте. Унаследован от "omjs.Widget.Button".

Добавляется на карту через "WidgetManager".

3.2.19 Класс AreaButton

Встроенный виджет, предоставляющий инструмент измерения площади на карте. Унаследован от "omjs.Widget.Button".

Добавляется на карту через "WidgetManager".

3.2.20 Класс Panel

Используется для добавления различных панелей на карту. Унаследован от "omjs.Widget".

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Widget.Panel**(<Object> *options*?)	omjs.Widget.Panel	Создаёт новый экземпляр виджета панели

Опции:

Опция	Тип	По умолчанию	Описание
content	String | DOMElement	' '	Содержимое панели

Методы:

Метод	Возвращает	Описание
getContentContainer()	jQuery HTMLElement	Возвращает DOM Элемент – контейнер контента панели

3.2.21 Класс Legend

Используется для отображения легенды картограммы. Унаследован от "omjs.Widget.Panel".

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Widget.Legend**(<Object> *options*?)	omjs.Widget.Legend	Создаёт новый экземпляр легенды

Опции:

Опция	Тип	По умолчанию	Описание
classificators	Array	[]	Объект содержимого легенды в формате: { 'name': '', // Подпись

			'fillColor': '#000000', // Цвет заливки 'strokeWidth': 1, // Толщина контура 'strokeColor': '#000000', //Цвет контура }
name	String	"	Заголовок панели легенды

Методы:

Метод	Возвращает	Описание
setClassificators(<Array> *список данных легенды*)	this	Устанавливает новый набор данных для легенды
setName(<String> *заголовок панели легенды*)	this	Устанавливает новый заголовок панели легенды

3.2.22 Класс Window

Базовый класс виджета окна с заголовком и нижней панелью. Унаследован от "omjs.Widget".

Используется для отображения большого объёма данных.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Widget.Window**(<Object> *options*?)	omjs.Widget.Window	Создаёт новый экземпляр виджета окна

Опции:

Опция	Тип	По умолчанию	Описание
showHeader	Boolean	true	Отображение заголовка окна
showFooter	Boolean	true	Отображение нижней панели окна
headerContent	String | DOMElement	''	Контент заголовка окна
footerContent	String | DOMElement	''	Контент нижней панели окна
content	String | DOMElement	''	Контент окна

beforeClose	Function	null	Функция, которая будет вызвана перед закрытием окна. Если функция вернёт false, окно не будет закрыто

Методы:

Метод	Возвращает	Описание
setHeader(<String|DOMElement> *контент заголовка*)	this	Устанавливает контент заголовка окна
setFooter(<String|DOMElement> *контент нижней панели*)	this	Устанавливает контент нижней панели окна
setContent(<String|DOMElement> *контент окна*)	this	Устанавливает контент окна

События:

Событие	Описание
close	Срабатывает по закрытию окна

3.2.23 Класс ZoomControl

Встроенный виджет масштабирования карты с указателем текущего увеличения (zoom). Унаследован от "omjs.Widget.ButtonGroup".

Вводится на карту через "WidgetManager".

3.2.24 Класс LayerSwitcher

Встроенный виджет для переключения подложки карты. Унаследован от "omjs.Widget.Button".

Вводится на карту через "WidgetManager".

3.2.25 Класс SidePanel

Виджет-панель. Унаследован от "omjs.Widget".

Используется для добавления панели, которая будет отображена рядом с картой.

Инициализация:

Фабрика/Конструктор	**Возвращает**	**Описание**
new **omjs.Widget.SidePanel(** <Object> *options*?)	omjs.Widget.SidePanel	Создаёт новый экземпляр виджета-панели. В объекте опций можно передать позицию, которая укажет, где нужно будет отобразить панель

Опции:

Опция	**Тип**	**По умолчанию**	**Описание**
position	String	'left'	Позиция панели. Принимает следующие значения: 'left', 'top', 'right', 'bottom'

3.2.26 Класс LayersTree

Виджет для отображения слоёв опубликованной карты. Унаследован от "omjs.Widget.SidePanel".

При добавлении виджета на карту на ней будут отображены все слои из виджета "Список слоёв".

Инициализация:

Фабрика/Конструктор	**Возвращает**	**Описание**
new **omjs.Widget.LayersTree(** <Object> *sidepanel options*?)	omjs.Widget.LayersTree	Создаёт новый экземпляр виджета-дерева опубликованных слоёв

3.2.27 Набор методов Event methods

Набор методов, которые есть у классов, поддерживающих события.

События обеспечивают вызов определённой функции при выполнении процедур с объектом, например, при выборе (клике) пользователем полигона на карте у объекта векторного слоя срабатывает событие "click":

```
Layer.on('click', function(e) {
    alert(e.olEvent.coordinate);
});
```

Функции в событиях передаются по ссылке. Для отвязывания функции от события необходимо определить функцию до подписки:

```
function onClick(e) { ... }
Layer.on('click', onClick);
Layer.off('click', onClick);
```

Методы:

Метод	**Возвращает**	**Описание**
addEventListener(<String> *type*, <Function> *fn*, <Object> *context?*)	this	Добавляет функцию, которая сработает при определённом событии объекта. Возможно опционально указать контекст для функции (определить "this"). Возможно указать несколько типов событий через пробел (например, 'click dblclick')
addOneTimeEventListener (<String> *type*, <Function> *fn*, <Object> *context?*)	this	Функционал аналогичен методу "addEventListener", но функция будет отписана от события сразу же после первого срабатывания
addEventListener(<Object> *eventMap*, <Object> *context?*)	this	Подписывает сразу несколько функций на различные события *{click: onClick, mousemove: onMouseMove}*
removeEventListener(<String> *type*, <Function> *fn?*, <Object> *context?*)	this	Отписывает указанную функцию от события. Если функция не указана – будут отписаны все функции от указанного события
removeEventListener(<Object> *eventMap*, <Object> *context?*)	this	Отписывает несколько функций от различных событий
removeEventListener()	this	Отписывает все функции от всех событий
hasEventListeners(<String> *type*)	Boolean	Возвращает "true", если на указанное событие существуют подписанные функции
fireEvent(<String> *type*, <Object> *data?*)	this	Вызывает событие указанного типа. Возможно передать объект с указанными данными: первый аргумент функции, подписанной на это событие, будет содержать в себе свойства этого объекта

clearAllEventListeners()	this	Отписывает все функции от всех событий
on(…)	this	Ссылка на метод "addEventListener"
once(…)	this	Ссылка на метод "addOneTimeEventListener"
off(…)	this	Ссылка на метод "removeEventListener"
fire(…)	this	Ссылка на метод "fireEvent"

3.2.28 Класс Class

Базовый класс, реализующий ООП. Почти все классы OMJS унаследованы от этого класса.

Класс обеспечивает простое наследование и предоставляет несколько специальных свойств: "options", "includes" и "statics".

Для создания нового класса используется "omjs.Class.extend":

```
var MyClass = omjs.Class.extend({
    initialize: function (greeter) {
        this.greeter = greeter;
        // конструктор класса
    },
    greet: function (name) {
        alert(this.greeter + ', ' + name)
    }
});
// Создаётся экземпляр класса MyClass, передавая строку "Hello" в конструктор
var a = new MyClass("Hello");
// Вызывается метод "greet", который вызовет алерт с текстом "Hello, World"
a.greet("World");
```

Метод "initialize" является конструктором класса и вызывается при выполнении "new MyClass(...)".

Также метод используется, чтобы наследовать от других указанных классов:

```
var MyChildClass = MyClass.extend({
    // ... новые свойства и методы
```

```
});
```

3.2.29 Класс Model

Базовый класс, представляющий модель данных.

Основная идея модели – разделение данных и логики. Данные хранятся внутри модели, как в базе данных; методы и свойства класса реализуют бизнес-логику и работу с этими данными.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Model**(<Object> *attributes*, <Object> *options*?)	omjs.Model	Создаёт новый экземпляр модели. Принимает объект с атрибутами (любые данные). Опции могут быть использованы дочерними классами

Методы:

Метод	Возвра-щает	Описание
set(<String> *ключ*, <String> *значение*)	null	Устанавливает новое значение в модель, которое будет доступно по заданному ключу
set(<Object> *объект значений*)	null	Устанавливает набор значений в модель. *{attr1:'value1', attr2:'value2',...}*
get(<String> *ключ значения*)	Any	Возвращает значение атрибута по ключу

3.2.30 Класс Collection

Базовый класс-коллекция, реализующий групповые методы работы с объектами/моделями.

Инициализация:

Фабрика/Конструктор	Возвращает	Описание
new **omjs.Collection**(<Array> *objects*)	omjs.Collection	Создаёт новую коллекцию с заданными объектами

Методы:

Метод	Возвра-щает	Описание
getLength()	Number	Длина коллекции
getArray()	Array	Элементы коллекции в виде массива
indexOf(<object> *объект*)	Number	Индекс элемента в коллекции, указанного в параметре object. Если элемент отсутствует в коллекции – метод вернет "-1"
has(<Object|Object[]> *объект[ы]*)	Boolean	Определение наличия элемента (массива элементов) в коллекции
at(<Number> *Index*)	Object	Элемент коллекции с указанным индексом
add(<Object|Object[]> *объект[ы]*)	this	Добавление элемента или массива элементов в коллекцию
remove(<Object|Object[]> *объект[ы]*)	this	Удаление элемента (элементов) из коллекции
pluck(<String> *имя атрибута*)	Array	Создание массива из значений переданного атрибута моделей коллекции
filter(<Function(obj, index)> *фильтр*)	omjs.Collection	Фильтрация элементов коллекции и получение их в виде новой коллекции. Принимает функцию, в которую будет передан объект и индекс. Функция должна вернуть Boolean (попадает объект под фильтр или нет)
each(<Function(obj, index)> *итератор*)	this	Цикл по коллекции. Функция-итератор будет вызвана для каждого объекта коллекции
reset(<Object[]> *объекты*)	this	Очищает коллекцию и записывает в нее элементы "objects"

3.2.31 Libraries

Библиотека **jQuery** входит в состав "omjs" и доступна через переменную "omjs.$".

Библиотека **OpenLayers** входит в состав "omjs" и доступна через переменную "omjs.ol".

3.3 Настройка работы программы через Extension API

Extension API использует плагины, обеспечивающие следующие процедуры работы:

− используя предобработчик, выполнить произвольный код перед требуемым методом. Предобработчик получает те же входные данные, что и исходный метод. Его основная задача – адаптировать их и выдать основному обработчику. Правило именования: должен иметь имя "before_<имя метода>";

− используя заменитель, выполнить произвольный код вместо требуемого метода. Должен иметь те же входные и выходные параметры (по количеству и типам), что и метод. Правило именования: должен иметь имя "instead_<имя метода>";

− используя постобработчик, выполнить произвольный код после требуемого метода в качестве параметров получает результат метода, а также его исходные параметры. В качестве конечного результата используется именно результат постобработчика. Правило именования: должен иметь имя "after_<имя метода>".

Для примера работы набора рассмотрим использование Extension API для генерации PDF-документа.

Задача:

Сформировать PDF-документ в соответствии с макетом.

Что заставило использовать расширение:

Базовое поведение системы предполагает, что в генерируемый PDF-документ будут попадать все заполненные значения объекта с учетом локализации, а значения группируются в блоки информации по заголовкам, которым они принадлежат. В утвержденном макете требовалось, чтобы выводились не все поля блока, а только определенные, некоторые значения, объединенные одним заголовком, должны попадать в другой блок информации, некоторые заголовки со всеми значениями не должны попадать в генерируемый файл.

Реализация:

При получении запроса на информацию по объекту в формате PDF Программа выполняет следующую последовательность действий:

- формирует экземпляр класса "Object";
- выполняет выборку этого объекта из СУБД (Object.SelectByID(<int: object_id>));
- выполняет сериализацию объекта (Object.FormatPdfApi()):
    1) создается экземпляр объекта класса "Pdf";
    2) выполняется подготовка данных объекта (значений полей объекта) посредством вызова "Object._prepareColumnsToPdf()".
- выполняется добавление контента в объект Pdf (Pdf.AddContent());
- сформированный PDF возвращается в виде ответа клиенту, который запросил информацию.

Для изменения фильтрации (какие значения должны попасть в итоговый документ) и группировки (какому заголовку должно принадлежать какое значение) необходимо изменить базовое поведение метода "_prepareColumnsToPdf()".

Для изменения базового поведения создан файл "plugins/v_2_0/Object.py". В созданном файле определен метод "instead__prepareColumnsToPdf()" – данный метод выполняется вместо метода "_prepareColumnsToPdf()". В методе выполняется код, который выбирает только те значения, которые необходимы для формирования PDF-файла. В созданном файле определен метод "after__prepareColumnsToPdf()", который выполнится после "instead__prepareColumnsToPdf()" и перегруппирует уже полученные значения в соответствии с утвержденным макетом.

3.4 Настройка работы программы через OMJS

3.4.1 Отображение опубликованной карты в соответствии с настройками публикации

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" https://example.com/static/omjs/2.8/omjs.css" />
  <script src="https://exemple.com/static/omjs/2.8/omjs.js"></script>
  <style>
    html, body, #map {
      width: 100%;
      height: 100%;
      margin: 0;
    }
  </style>
</head>
<body>
  <div id="map" />
  <script>
    omjs.initView('#map',
'https://example.com/public_map/{project}/{map}/config.json')
      .done(view => {
        view.showGisPublication(); // Показ опубликованной карты
      });
  </script>
</body>
</html>
```

3.4.2 Использование экземпляра View для управления картой

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="https://example.com//static/omjs/2.8/omjs.css" />
  <script src="https://example.com//static/omjs/2.8/omjs.js"></script>
  <style>
    html, body, #map {
      width: 100%;
      height: 100%;
      margin: 0;
    }
```

```
    </style>
  </head>
  <body>
    <div id="map" />
    <script>
      omjs.initView('#map',
'https://example.com//public_map/{project}/{map}/config.json')
        .done(view => {
          view.showGisPublication();

          // Переключение подложки
          view.switchBaseLayer(4441); // Id подложки можно узнать в
config.json

          // Изменение центра и масштаба карты
          const olView = view.getMap().getView();
          olView.setZoom(10);
          olView.setCenter(omjs.ol.proj.fromLonLat([ 30.3141300,
59.9386300 ]));
        });
    </script>
  </body>
  </html>
```

3.4.3 Добавление слоёв на карту по их коду

```
  <!DOCTYPE html>
  <html>
  <head>
    <link rel="stylesheet"
href="https://example.com/static/omjs/2.8/omjs.css" />
    <script src="https://example.com/static/omjs/2.8/omjs.js"></script>
    <style>
      html, body, #map {
        width: 100%;
        height: 100%;
        margin: 0;
      }
    </style>
  </head>
  <body>
    <div id="map" />
    <script>
```

```
        omjs.initView('#map',
'https://example.com/public_map/{project}/{map}/config.json')
            .done(view => {
                view.getContainer()[0].style.opacity = 1; // Принудительное
отображение карты без публикации

                view.addLayer('layer_code1 layer_code2'); // Отображение
растровых слоёв

                view.addLayer('ayer_code3', { layerType: 'vector' }) //
Отображение векторного слоя
                    .done(layer => {
                        // Устанавка стилей векторного слоя
                        const s = omjs.ol.style;
                        const stroke =  new s.Stroke({ color: '#f00', width: 1 });
                        const fill = new s.Fill({ color: '#f005' });

                        layer.setStyle(
                            new s.Style({
                                image: new s.Circle({ fill, stroke, radius: 12 }),
                                stroke,
                                fill
                            })
                        );
                    });
            });
        </script>
    </body>
    </html>
```

3.4.4 Обработка событий мыши и отображение балуна на карте

```
    <!DOCTYPE html>
    <html>
    <head>
        <link rel="stylesheet"
href="https://example.com/static/omjs/2.8/omjs.css" />
        <script src="https://example.com/static/omjs/2.8/omjs.js"></script>
        <style>
            html, body, #map {
                width: 100%;
                height: 100%;
                margin: 0;
            }
        </style>
```

```
    </head>
    <body>
      <div id="map" />
      <script>
        omjs.initView('#map',
'https://example.com/public_map/{project}/{map}/config.json')
          .done(view => {
            view.showGisPublication()
              .done(() => {
                view.getLayers().each(layer => {
                  layer.on('click', e => {
                    const feature = e.feature.get('features')
                      ? e.feature.get('features')[0]
                      : e.feature

                    view.showBalloon({
                      content: 'Object id: ' + feature.get('obj').id +
' ',
                      position:
omjs.ol.proj.toLonLat(e.olEvent.coordinate)
                    })
                  });
                });
              });
          });
      </script>
    </body>
    </html>
```

3.4.5 Отображение встроенных виджетов

```
    <!DOCTYPE html>
    <html>
    <head>
      <link rel="stylesheet"
href="https://example.com/static/omjs/2.8/omjs.css" />
      <script src="https://example.com/static/omjs/2.8/omjs.js"></script>
      <style>
        html, body, #map {
          width: 100%;
          height: 100%;
          margin: 0;
          overflow: hidden;
        }
```

```
            .switcher {
               top: 16px;
               right:  16px;
            }

            .zoom {
               top: 16px;
               left: 16px;
            }

            .info {
               top: 16px;
               left: 64px;
               background-image:
url(
ADgdz34AAAABmJLR0QA/wD/AP+gvaeTAAABZElEQVRIibXVO0sdQRjG8Z+XI
0oQJF4bSS1qmegX8DMogqVoI/gZ7DUkXZpA2gSCnZCvYIqIWtl4QUULb2mCAY
/FjnBcZy+e1QcG9p153+e/s7MzwyurJRXX8AW3WGzo78B7vAvxATZD3rP0DXW
shLgfn3AV+hvbFT6ir6z5XCj8KpnZJE5xh19YwHIEdIKJIvM3IXEPXRjHDc4xlcpN
A+q4xmgeYD4kTqMNO/iLsUhuDFDHVqiN6jP2JYs8GwqWMnKzAHXMZAHa0R
2e1yVT7mwC8DML0KgjbOSM5wEO08mtEYNByYI3o6EygBp6MgyK/vlauiO9k0m
mCrv4lxobxkABpCUzSAGa1SPP2Cd6UcUA/yv4PamNAc4qAE7LAH5XAGyWAXy
vAPhRJqkN2/J3bKz9yXjhqEbEL5msdqPguI5pQnJkFJkf48NzzR/UizVcRowvsIq3e
QaxnRxTTfzSr7JnXkb3h3mFPyL4H78AAAAASUVORK5CYII=);
               background-repeat: no-repeat;
               background-position: center;
            }
         </style>
      </head>
      <body>
         <div id="map" />
         <script>
            omjs.initView('#map',
'https://example.com/public_map/{project}/{map}/config.json')
               .done(view => {
                  view.showGisPublication();

                  const widgets = view.getWidgetManager();
                  widgets.addWidget('LayerSwitcher', { classes: 'switcher' });
                  widgets.addWidget('ZoomControl', { classes: 'zoom' });

                  // Создание виджета вручную
                  const info = new omjs.Widget.Button({ classes: 'info' });
                  info.on('click', () => alert('Произошло нажатие.'));
                  widgets.addWidget(info);
```

```
            });
        </script>
    </body>
    </html>
```

3.4.6 Загрузка данных объекта слоя по клику на карте

```
    <!DOCTYPE html>
    <html>
    <head>
        <link rel="stylesheet"
href="https://example.com/static/omjs/2.8/omjs.css" />
        <script src="https://example.com/static/omjs/2.8/omjs.js"></script>
        <style>
            html, body, #map {
                width: 100%;
                height: 100%;
                margin: 0;
            }
        </style>
    </head>
    <body>
        <div id="map" />
        <script>
            omjs.initView('#map',
'https://example.com/public_map/{project}/{map}/config.json')
                .done(view => {
                    const project = view.getProject(); // Получение проекта

                    view.showGisPublication()
                        .done(() => {
                            view.getLayers().each(layer => { // Получение слоёв
                                layer.on('click', e => { // Показ информации по
объекту при клике на него
                                    const projLayer = layer.getProjectLayers?.().at(0); //
Получение слоя из проекта

                                    if (projLayer?.objects && !e.feature.get('features')) {
                                        project.loadObject( // Загрузка данных объекта
                                            projLayer.get('code'),
                                            e.feature.get('obj').id
                                        )
                                        .then(obj => {
                                            const fields = obj._attributes;
                                            view.showBalloon({
```

```
                                content: Object.keys(fields).map(k => k + ': ' +
fields[k]).join('<br/>'),
                                position:
omjs.ol.proj.toLonLat(e.olEvent.coordinate)
                            });
                        });
                    }
                });
            });
        });
    </script>
</body>
</html>
```

3.4.7 Работа с данными без отображения карты

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://example.com/static/omjs/2.8/omjs.js"></script>
    <style>
        body {
            width: 100%;
            margin: 0;
            padding: 16px;
            box-sizing: border-box;
        }

        table {
            width: 100%;
            border-collapse: collapse;
        }

        td, th {
            border: 1px solid #ddd;
            padding: 8px 16px;
        }

        th {
            background: #eee;
        }
    </style>
</head>
<body>
```

```
<script>
  omjs.initProject({ // Загрузка проекта без View
    mapCode: '{map}',
    apiUrl: 'https://example.com/api/2.8/{project}/'
  })
    .then(proj => {
      proj.loadLayers()
        .then(layers => {
          let projLayer = layers.at(0);
          projLayer.loadObjects({ fields: ['*'] }) // Загрузка всех
полей слоя
            .then(() => {
              proj.loadStructure(projLayer.get('code')) // Загрузка
структуры слоя
                .then(structure => {
                  let html = '';
                  const struct = structure.filter(s => ![ 'gis_id',
'geom' ].includes(s.code));
                  const keys = struct.map(s => s.code);

                  html += `
                    <h2>${projLayer.get('name') ||
projLayer.get('code')}</h2>
                    <table>
                      <thead>
                        ${keys.map(k => `<th>${struct.find(s =>
s.code === k).name || k}</th>`).join('')}
                      </thead>
                      <tbody>`;
                  projLayer.objects.each(obj => {
                    html += `<tr>${keys.map(k =>
`<td>${obj.get(k)}</td>`).join('')}</tr>`;
                  });
                  html += '</tbody></table>'
                  document.body.innerHTML = html;
                });
            });
        })
    });
</script>
</body>
</html>
```

4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1 Входные данные

На вход в Программу подаются следующие типы данных:

− указанные пользователем геопространственные данные;

− геопространственные данные из хранилищ СУБД ПО «Ситроникс.ГИС»;

− геопространственные данные, предоставленные по API из сторонних приложений;

− геопространственные данные из импортированных файлов.

Кодирование входных данных выполняется в соответствии с протоколами обмена.

4.1.1 Формат и описание входных данных

Программа имеет интерфейс, позволяющий загружать (импортировать) векторные и растровые данные в следующих форматах:

− форматы векторных геоданных (в скобках указаны расширения импортируемых файлов):

1) ESRI Shapefile (.shp);
2) ESRI Geodatabase multiple (.gdb);
3) Google KML (.kml, .kmz);
4) GeoJSON (.json, .geojson);
5) AutoCAD DXF (.dxf);
6) Panorama SXF (.sxf);
7) MapInfo File (.mid/.mif, .tab);
8) формат хранения и обмена данными стандарта Global Positioning System – GPS (.gpx);
9) формат системы автоматизированного проектирования (далее – САПР) MicroStation версии 7 и выше (.dgn).

Примечание – Формат shp имеет следующие ограничения:

− Единовременно может храниться только 1 тип объектов (точка, линия или полигон);

− Заголовки (имена полей) обрезаются до 10 символов (до 5 символов на русском языке);

− Размер файла не может превышать 2 Гб.

Примечание – В файлах формата tab единовременно может храниться несколько объектов (точка/линия/полигон).

Примечание – Данные в форматах shp, mif, tab и sxf корректно отрабатываются и импортируются в систему только в виде сформированных архивов, содержащих необходимые служебные файлы (в соответствии со спецификацией формата).

− форматы растровых геоданных (в скобках указаны расширения импортируемых файлов):

1) GeoTIFF (.tiff, .tif, .geotiff, .geotif);
2) JPEG (.jpg, .jpeg);
3) Portable Network Graphics (.png).

− форматы данных без геометрических характеристик (в скобках указаны расширения импортируемых файлов):

1) Microsoft Office Excel (.xls, .xlsx);
2) Comma Separated Value (.csv).

Описание работы с импортированными данными приведено в документе «Программное обеспечение «Ситроникс.ГИС». Руководство системного программиста».

4.2 Выходные данные

4.2.1 Модуль управления доступом

Выходные данные Программы содержат пространственную, атрибутивную, аналитическую, сопроводительную и другую информацию в следующих формах представления:

- экспорт файлов;
- визуализация геоданных;
- аналитическая отчётность;
- публикация карт (слоёв);
- стилизация элементов;
- локализация языкового отображения элементов.

Формы предоставления выходных данных: векторные ГИС-слои; электронные таблицы; аналитические отчеты, метаданные.

Расчётные операции для формирования выходных данных из Программы выполняет серверное ПО.

Визуализация (отображение) выходных данных из Программы обеспечивается клиентским ПО Программы или другими программными продуктами, поддерживающими обмен данными по API.

4.2.2 Формат, описание и способ кодирования выходных данных

Программа имеет интерфейс, позволяющий передавать (экспортировать) векторные и растровые данные сторонним программным приложениям.

Экспорт данных возможен в следующих форматах (в скобках указаны расширения экспортируемых файлов):

- ESRI Shapefile (.shp);
- ESRI Geodatabase multiple (.gdb);
- Google KML (.kml, .kmz);
- GeoJSON (.json, .geojson);
- MapInfo File (.mid/.mif, .tab);
- GeoTIFF (.geotiff, .geotif);
- Portable Network Graphics (.png);
- Microsoft Office Excel (.xls, .xlsx);
- Comma Separated Value (.csv).

5 СООБЩЕНИЯ

Сообщения программисту включают в себя следующие категории:

- сообщения о доступности данных;
- сообщения о выполнении операций;
- сообщения о готовности / неготовности.

При появлении проблем в работе Программы рекомендуется обратиться в службу технической поддержки.