**ООО «АМ-ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ»**

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |

**Система МОНИТОРИНГА Парковочных мест**

**АРТИКУЛ**

**Описание ПРОГРАММЫ**

0000000

Москва 2023

АНнотация

Настоящий документ содержит сведения о функциональном назначении, структуре программного обеспечения «Система мониторинга парковочных мест», артикул , условиях его функционирования, способах доступа к интерфейсу и API системы, описание входных и выходных данных.

Настоящий документ составлен в соответствии с требованиями ГОСТ 19.402‑78 и   
ГОСТ 19.106‑78.

Содержание

[1. Общие сведения 5](#_Toc138857093)

[1.1. Обозначение и наименование программы 5](#_Toc138857094)

[1.2. Краткое наименование 5](#_Toc138857095)

[1.3. Языки программирования, на которых написана программа 5](#_Toc138857096)

[2. Функциональное назначение 6](#_Toc138857097)

[3. Описание логической структуры 8](#_Toc138857098)

[3.1. Алгоритм программы 8](#_Toc138857099)

[4. Компонентой формирования отчетов выполняется периодическое формирование отчетов и рассылка их адресатам электронной почты, согласно настройкам ПО, заданным оператором. 9](#_Toc138857100)

[3.1.1. Алгоритмы авторизации и получения привилегий пользователя 9](#_Toc138857101)

[3.1.2. Алгоритмы создания/сохранения разметки парковочных мест 9](#_Toc138857102)

[3.1.3. Распознавания статуса занятости парковочных мест 10](#_Toc138857103)

[3.1.4. Методика определения точности распознавания статуса занятости парковочных мест 11](#_Toc138857104)

[3.1.5. Алгоритм поиска парковочных мест по УНОМ 11](#_Toc138857105)

[3.1.6. Алгоритм поиска парковочных мест, внутри заданной географическими координатами области 11](#_Toc138857106)

[3.1.7. Алгоритм опроса камер ЕЦХД для получения изображений с камер, наблюдаемых парковочных пространств 11](#_Toc138857107)

[3.1.8. Алгоритм формирования еженедельных отчетов 11](#_Toc138857108)

[3.1.9. Алгоритм отображения на карте парковочных мест и камер в виде точек с поддержкой кластеризации при масштабировании 11](#_Toc138857109)

[3.2. Программный интерфейс для внешних систем (API ПО) 12](#_Toc138857110)

[3.2.1. Коды ответов 12](#_Toc138857111)

[3.2.1. Объекты 13](#_Toc138857112)

[3.2.2. Методы программного интерфейса 13](#_Toc138857113)

[3.3. Используемые методы 14](#_Toc138857114)

[3.3.1. Объектно-ориентированный подход при описании задач 14](#_Toc138857115)

[3.3.2. Объектно-ориентированные языки разработки 15](#_Toc138857116)

[3.3.3. Объектно-реляционная база данных 16](#_Toc138857117)

[3.3.4. Стандартизация и унификация 17](#_Toc138857118)

[3.3.5. Использованные при разработке библиотеки 18](#_Toc138857119)

[3.4. Структура программы с описанием функций составных частей и связи между ними 18](#_Toc138857120)

[3.5. Функциональная структура ПО 18](#_Toc138857121)

[3.6. Информация о пользователях ПО, их привилегиях и разграничении доступа 18](#_Toc138857122)

[3.7. Задачи, решаемые графическим интерфейсом пользователя 19](#_Toc138857123)

[3.8. Задачи, решаемые подсистемой сбора данных 19](#_Toc138857124)

[3.9. Задачи, решаемые подсистемой анализа данных 19](#_Toc138857125)

[3.10. Задачи, решаемые подсистемой представления данных 19](#_Toc138857126)

[3.10.1. Задачи, решаемые компонентой формирования отчетов 19](#_Toc138857127)

[3.11. Задачи, решаемые подсистемой сбора параметров работы 19](#_Toc138857128)

[3.12. Средства и способы связи для информационного обмена между компонентами 19](#_Toc138857129)

[3.13. Связи ПО с другими программами 20](#_Toc138857130)

[4. Используемые технические средства 21](#_Toc138857131)

[5. Вызов и загрузка 23](#_Toc138857132)

[5.1. Способ вызова программы с соответствующего носителя данных 23](#_Toc138857133)

[5.2. Входные точки в программу 23](#_Toc138857134)

[6. Входные данные 24](#_Toc138857135)

[7. Выходные данные 26](#_Toc138857136)

[7.1.1. Структура данных 26](#_Toc138857137)

[7.1.2. Формат данных 26](#_Toc138857138)

[7.1.3. Описание данных 26](#_Toc138857139)

[ПЕРЕЧЕНЬ ТЕРМИНОВ 27](#_Toc138857140)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 28](#_Toc138857141)

# Общие сведения

## Обозначение и наименование программы

Полное наименование программы: Система мониторинга парковочных мест, артикул.

## Краткое наименование

Краткое наименование программы: ПАРКОВКИ.

## Языки программирования, на которых написана программа

Основным языком программирования является GoLang.

# Функциональное назначение

Программа предназначена для автоматизации процессов мониторинга парковочного пространства общего пользования г. Москвы, обеспечивающих реализацию следующих функций:

выполнение аутентификации субъектов доступа (посредством механизма ввода комбинации идентификатора учетной записи (логина) и пароля пользователя) через СУДИР;

выполнение авторизации субъектов доступа;

управление субъектами доступа;

управление правами субъектов доступа;

графический интерфейс пользователя Технологического портала, который состоит из набора интерактивных веб-страниц;

интерактивная карта с нанесенными на нее значками, обозначающими все доступные для ПО видеокамеры и зарегистрированные в ПО парковочные пространства;

интерфейс детализированного представления каждого парковочного пространства, где отображен последний снимок, полученный с выбранной Видеокамеры, с автоматически нанесенными на него областями проекций распознанных ПО парковочных мест и их текущим статусом в виде особой цветовой окраски для каждого и статусов (свободно/занято/статус неизвестен);

интерфейс интерактивного проектирования размещения парковочных мест на карте с привязкой к камере и регистрацией географических координат каждого парковочного места;

интерфейс реестра парковочных пространств и парковочных мест;

интерфейс интерактивного проектирования и отображения границ и географических координат парковочных мест каждого парковочного пространства, обеспечивающего возможность корректировки парковочных мест:

* + - * добавление парковочных мест;
      * задание границ парковочных мест;
      * удаления парковочных мест;
      * редактирования географических координат парковочных мест.

интерфейс создания задач для наблюдения за парковками с привязкой к одной из камер дворового наблюдения с настройкой позиции камеры для сбора данных и выбора периода анализа занятости мест;

интерфейс просмотра списка камер, изображений с камер для настройки мониторинга парковки;

взаимодействие со шлюзом ЕЦХД для получения информации окамерах и изображений с камер;

взаимодействие с Модулем распознавания регистрационных знаков и определения характеристик транспортных средств, для получения bbox запаркованных транспортных средств;

REST API, возвращающий по запросу данные по парковочным местам, доступным для Видеокамер, установленных по УНОМ;

REST API, возвращающий по запросу данные по парковочным местам, внутри заданной географическими координатами области;

формирование еженедельных отчетов о состоянии парковочного пространства;

логирование внутренних событий и результатов операций ПО в консоль.

# Описание логической структуры

## Алгоритм программы

Основной алгоритм выполнения программы состоит из следующих этапов:

1. Оператор ПО авторизуется через СУДИР и получает доступ к графическому интерфейсу пользователя. В графическом интерфейсе пользователя оператор ПО создает задания на наблюдение за парковочными местами. Для этого:

* Из списка камер дворового наблюдения г. Москвы, доступного оператору, выбирается камера;
* Для выбранной камеры создается парковочное пространство, наблюдаемое этой камерой;
* На изображении парковочного пространства, полученного с камеры, размечаются парковочные места, за которыми будет вестись наблюдение;
* На картографической основе отображается точка в координатах выбранной камеры и точки расположения парковочных мест, расположенных на парковочном пространстве, наблюдаемом выбранной камерой;
* Координатой парковочного места, является геометрический центр четырехугольника, определяющего границы парковочного места в проекции на картографическую основу;
* В интерактивном режиме на картографической основе можно отредактировать координаты парковочных мест;
* Задается интервал дат и временной диапазон наблюдения за выбранным парковочным пространством;
* Задается признак активности задания на мониторинг (вкл/выкл наблюдение);
* Информация о Задании и его настройках сохраняется в БД.

1. Компонента анализа данных циклически формирует список активных заданий наблюдения за парковочными пространствами, для каждого элемента этого списка выполняются операции:

* Для каждого наблюдаемого парковочного пространства с заданной частотой выполнится запрос изображения с камеры, заданной в настройке.
* Полученное изображение передается в детектор ТС.
* Для каждого переданного изображения детектор ТС возвращает массив bbox выявленных ТС.
* Полученный массив анализируется на предмет пересечения bbox ТС и границ расположения парковочных мест для данной камеры.
* Для каждого парковочного места вычисляется признак пусто/занято.
* Полученная информациям сохраняется в БД.

1. Актуальные данные о доступности парковочных мест могут быть получены через API или в графическом интерфейсе пользователя на интерактивной карте в виде точек с разной цвет графической схемой, соответствующей статусу свободно/занято для каждого парковочного места.

### Компонентой формирования отчетов выполняется периодическое формирование отчетов и рассылка их адресатам электронной почты, согласно настройкам ПО, заданным оператором.

### Алгоритмы авторизации и получения привилегий пользователя

Пользователь системы авторизуется через интерфейс авторизации СУДИР, для авторизованного в СУДИР ПО получает токен доступа JWT по которому через API СУДИР считываются ученые данные пользователя СУДИР. В БД ПО ищется пользователь с логином учетной записи СУДИР, если такой пользователь есть в БД, то для него получается из БД роль пользователя в ПО и согласно роли в ПО, предоставляется интерфейс пользователя ПО. Если пользователь в БД не найден, то создается новая запись с учётными данными СУДИР и назначается роль с минимальными правами в системе, только чтение.

Роль каждого пользователя, зарегистрированного в ПО, кроме администратора (обособленная учетная запись с максимальными правами и не требующая авторизации в СУДИР), может быть изменена администратором ПО.

### Алгоритмы создания/сохранения разметки парковочных мест



Разово-ручное полигональное выделение мест поверх трансляции с камеры.

При любом угле съемки парковочного места его можно описать четырехугольником.

Полигональная область парковки усложнит алгоритмы вычисления занятости и увеличит затраты, поэтому область парковки задается четырехугольником, скорее всего даже параллелограммом.

Возможные сценарии разметки парковочного пространства:

1. Парковочные места являются нумерованными, для каждого парковочного места с номером передаем номер и состояние "свободно/занято" во Внешнюю систему.
2. Существует территория, представляющая область интереса, для которой задается количество допустимых парковочных мест. Оператор не задает точное расположение парковочных мест. Определяется количество припаркованных автомобилей внутри области интереса, передается информация, что на определенной территории присутствуют 8 авто из 10 возможных для этой территории.

Для сценария №1 в интерфейсе пользователя выполняются следующие операции:

### Алгоритм определения статуса занятости парковочных мест



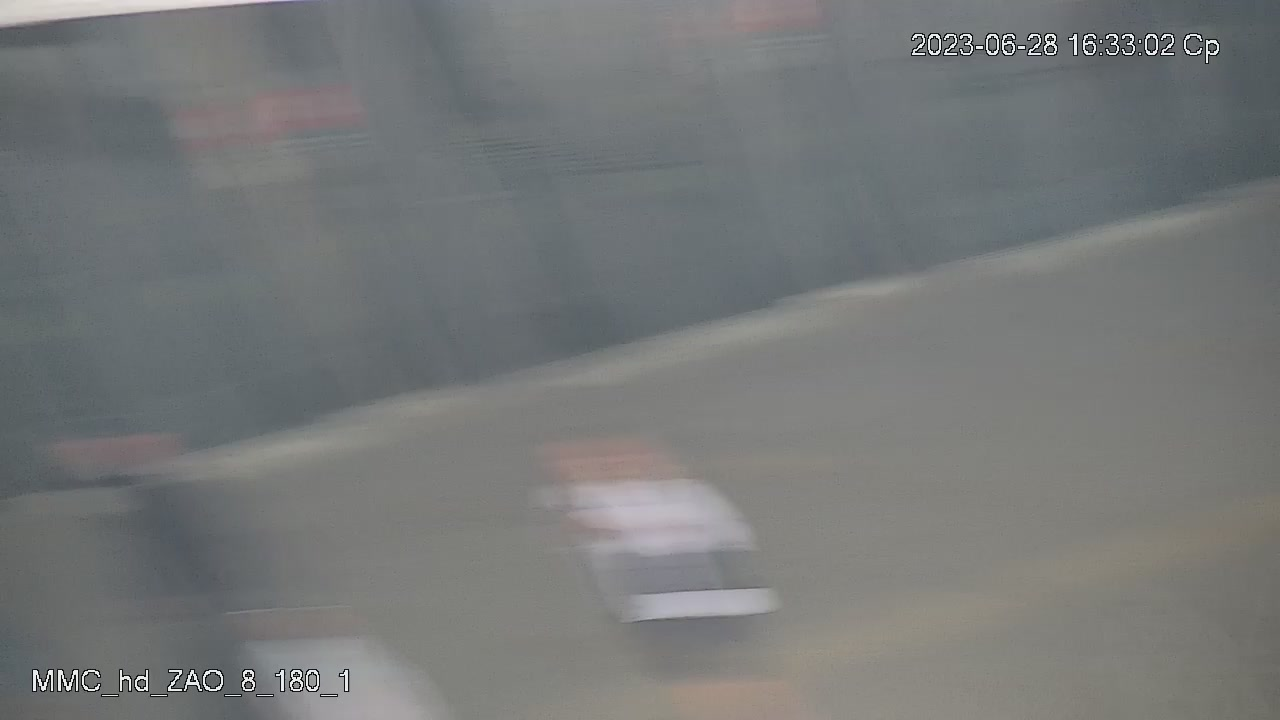


### Методика определения точности распознавания статуса занятости парковочных мест

НФТ: Методика должна быть описана в Регламенте оказания услуг.

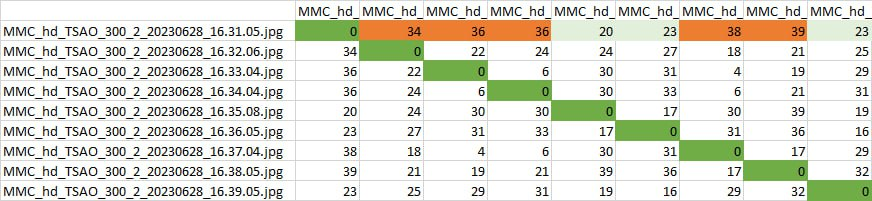
### Алгоритм определения соответствия позиции камеры, заданной при получении изображения

Поворотные камеры ЕЦХД, выполняют операции позиционирования, проверки позиции и получения изображения с камеры, отдельными операциями через REST API шлюза ЕЦХД. Также к каждой камере городского наблюдения может выполняться одновременное обращение нескольких сторонних приложений через шлюз ЕЦХД. В связи с этим возможны ситуации, когда компонента сбора данных ПО будет получать изображение с камеры, находящейся не в той позиции, которая задана в настройках задания мониторинга парковочного пространства. Также возможно получение изображения с камеры, находящейся в движении. Пример:



Для предотвращения анализа изображений с камер из несоответствующих настройкам положений, выполняются следующие действия проверки достоверности получаемых изображений:

* Выполняется проверка текущей позиции камеры
* Если текущая позиция камеры не соответствует заданной в настройке задания, выполняется смена позиции камеры
* Выполняется скачивание изображения по ссылке с камеры
* Выполняется проверка текущей позиции камеры
* Если текущая позиция камеры не соответствует заданной в настройке задания, полученное изображение считается ложным
* С полученного изображения вычисляется хеш значение по алгоритму из методов библиотеки opencv (https://habr.com/ru/articles/519454/), и сравнивается с предыдущими десятью вычисленными хеш значениями для данной камеры, если отклонение значения хеш не превышает 15%, то изображение считается достоверным. Пример:



### Алгоритм поиска парковочных мест по УНОМ

### Алгоритм поиска парковочных мест, внутри заданной географическими координатами области

НФТ: Время отклика Решения на запросы внешних ИСиР не должно превышать 1 (одной) секунды.

### Алгоритм опроса камер ЕЦХД для получения изображений с камер, наблюдаемых парковочных пространств

НФТ: Частота опроса Видеокамер — один раз в 30 (тридцать) секунд.

### Алгоритм формирования еженедельных отчетов

### Алгоритм отображения на карте парковочных мест и камер в виде точек с поддержкой кластеризации при масштабировании

## Программный интерфейс для внешних систем (API ПО)

Поддерживаются представленные ниже методы API (таблица 3).

Таблица 3 – Методы API

| МЕТОД | ВЫПОЛНЯЕМОЕ ДЕЙСТВИЕ | ПОДСИСТЕМА / МОДУЛЬ |
| --- | --- | --- |
| **Получение данных по УНОМ** | | |
| GET/ | запрос с параметрами УНОМ | Подсистема сбора и обработки данных |
| **Получение данных по географическим координатам** | | |
| POST / | запрос с параметрами прямоугольной ограничивающей области с координатами | Подсистема сбора и обработки данных |

### Коды ответов

ПО возвращает корректные коды ответа HTTP в зависимости от обработки запроса (таблица 4).

Таблица 4 – Коды ответов

| КОД ОТВЕТА HTTP | НАЗНАЧЕНИЕ |
| --- | --- |
| 200 OK | Запрос успешно обработан.  Возвращается для запросов на получение информации: поиск, получение списков и отдельных записей |
| 201 Created | Объект успешно создан.  Возвращается для запросов на создание объектов |
| 400 Bad Request | Запрос не соответствует формату.  Возвращается для запросов, которые не могут быть обработаны из-за несоответствия формату взаимодействия |
| 401 Unauthorized | Запрос не авторизован.  Возвращается при попытке выполнения запроса без аутентификации или с неверной аутентификационной информацией |
| 403 Forbidden | Объект, на который ссылается запрос, не существует. |
| 404 Not Found | Объект, на который ссылается запрос, не существует.  Возвращается для запросов, имеющих ссылку на объект по идентификатору в случае отсутствия объекта с указанным идентификатором |
| 409 Conflict | Создаваемый объект существует.  Возвращается для запроса на создание объекта, который уже был создан ранее |
| 500 Internal Server Error | Ошибка ПО  Возвращается во всех случаях ошибки при выполнении запроса, если ошибка не позволила обработать запрос |

### Объекты

#### Bounding Box

Bounding Box, координаты– прямоугольник, который задаётся координатами левого верхнего угла и правого нижнего.

Пример JSON:

{

"top": 100,

"left": 20,

"bottom": 340,

"right": 260

}

### Методы программного интерфейса

##### Получение данных по УНОМ

Запрос:

GET /:id

id – идентификатор **УНОМ**.

Ответ: возвращает объект <ххх>

Массив Парковочных мест, содержащий:

* Статус парковочного места:
  + место свободно
  + место занято
  + статус неизвестен
* Место для инвалидов (да/нет)
* Место спецтехники (да/нет)
* Координаты парковочного места

##### Получение данных по географическим координатам

Запрос:

POST /

запрос с параметрами прямоугольной ограничивающей области с координатами,

Тело запроса представляет собой JSON вида:

{

• Южная координата широты ограничивающей области;

• Западная координата долготы ограничивающей области;

• Северная координата широты ограничивающей области;

• Восточная координата долготы ограничивающей области.

}

Ответ: возвращает объект <ххх>

Массив Парковочных мест, содержащий:

* Статус парковочного места
  + место свободно
  + место занято
  + статус неизвестен
* Место для инвалидов (да/нет)
* Место спецтехники (да/нет)
* Координаты парковочного места

## Используемые методы

### Объектно-ориентированный подход при описании задач

Решение задач опирается на применение объектно-ориентированной технологии их описания и реализации. Основой этой технологии служит объектная модель ПО.

Объектная модель ПО описывает структуру объектов, составляющих ПО, их свойства, операции и взаимосвязи с другими объектами.

В объектной модели отражены те понятия и объекты реального мира, которые важны для разработки. В ней отражается прежде всего прагматика разрабатываемой системы. Прагматика выражается в использовании терминологии прикладной области, связанной с использованием разрабатываемой системы.

Объектом называется понятие, абстракция или любая другая вещь с четко очерченными границами, имеющая смысл в контексте рассматриваемой прикладной проблемы.

Введение объектов преследует две цели:

* понимание прикладной задачи (проблемы);
* введение основы для реализации ее на компьютере.

Целью разработки объектной модели является выделение и описание объектов, составляющих в совокупности проектируемую систему, а также выявление и указание различных зависимостей между объектами.

### Объектно-ориентированные языки разработки

При разработке использовался современный объектно-ориентированный язык программирования GoLang.

GoLang — это многопоточный компилируемый язык (интерпретатор в экосистеме также предусмотрен, но на практике необходимости в нем нет: компиляция происходит мгновенно). По производительности Go почти не уступает C++ и в десятки раз превосходит скриптовые языки — такие, как JavaScript, Python, Ruby, PHP.. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы.

Низкая требовательность к памяти. В GoLang есть «сборщик мусора» — периодически те объекты, потребность в которых уже не возникнет, удаляются. Таким образом происходит автоматическая очистка памяти.

Простота параллельных вычислений. Язык идеально подходит для создания программ, рассчитанных на многоядерные процессоры. Многопоточность, реализуемая через так называемые горутины (go routines), которые взаимодействуют посредством каналов, позволяет вызывать несколько функций практически одновременно. Эта возможность очень актуальна при создании больших и сложных программ. Средства параллельного программирования экономят время разработчика и помогают равномерно распределять ресурсы процессора.

### Объектно-реляционная база данных

База данных выступает в качестве информационной модели предметной области, в рамках которой функционирует система.

В БД отражается специфика предметной области, особенности решаемых на ее основе задач.

Базы данных предназначены для того, чтобы обеспечивать:

* учет потребностей всех решаемых задач, моделей, всех пользователей на данном уровне управления;
* однозначность представления и понимания используемой информации на разных уровнях и объектах (за счет использования единых и согласованных словарей и классификаторов);
* адекватное отображение предметной области, но, вместе с тем, состав информации на каждом уровне управления не должен превышать его потребностей, а быть достаточным и обобщенным в необходимой степени;
* хранение (в случае необходимости):

1. картографической информации;
2. фактографической информации;
3. документальной информации;
4. мультимедийной информации;

* обслуживание пользователей за время не более допустимого с возможными дополнительными требованиями на обслуживание заданного количества пользователей;
* принцип открытости, т. е. БД должна обладать информационной полнотой и, в то же время, быть способной к структурному информационному расширению при появлении новых категорий пользователей или прикладных задач;
* по возможности соответствие реально существующим связям и объектам предметной области для уменьшения вероятности программных ошибок, связанных с неправильной интерпретацией данных, а также облегчения разработки средств ведения БД.

В качестве СУБД, отвечающей всем перечисленным выше требованиям выбрана СУБД «Квант-Гибрид».

СУБД «Квант-Гибрид»– свободная объектно-реляционная система управления базами данных.

СУБД «Квант-Гибрид» поддерживает язык SQL и отвечает основным требованиям стандарта SQL:2011.

Работа с данными, хранящимися в БД, ведется на основе языка структурированных запросов SQL.

SQL (англ. Structured Query Language – «язык структурированных запросов») – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

SQL является информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных.

При всех изменениях SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных.

### Стандартизация и унификация

Методы программных интерфейсов Системы (API) спроектированы с учетом требований унификации:

* для выполнения аналогичных операций должны использоваться аналогичные методы;
* должны быть унифицированы термины, используемые для описания идентичных понятий, операций и действий пользователя;
* реакция системы на действия пользователя (посредством API-ответов) должна быть типовой для каждого действия.

Использованное при создании Системы программное обеспечение и библиотеки программных кодов являются общедоступными и используются в промышленных масштабах.

### Использованные при разработке библиотеки

При разработке использованы следующие библиотеки (приведены со ссылками на их страницы в сети Интернет):

* СУБД «Квант-Гибрид» 1.4. «https://granit-concern.ru/products/informatsionnye-sistemy/informatsionnye-sistemy\_5.html»;
* Операционная система MOS. «https://os.mos.ru/git/MOS»

Описать:

1. React-js

2. Go-lang

3. Kafka

4. Nginx

5. Docker

6. Картографическая основа 2Гис

## Структура программы с описанием функций составных частей и связи между ними

Общая архитектура ПО представлена ниже (рисунок 8).

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 8 – Общая архитектура ПО

## Функциональная структура ПО

Структурно ПО состоит из следующих подсистем:

* Компонента анализа
* Компонента сбора данных
* Компонента представления данных
* Компонента формирования отчетов
* Компонента сбора параметров работы
* База данных парковок и статистики доступности
* Графический интерфейс пользователя

## Информация о пользователях ПО, их привилегиях и разграничении доступа

## Задачи, решаемые графическим интерфейсом пользователя

## Задачи, решаемые подсистемой сбора данных

## Задачи, решаемые подсистемой анализа данных

## Задачи, решаемые подсистемой представления данных

### Задачи, решаемые компонентой формирования отчетов

## Задачи, решаемые подсистемой сбора параметров работы

## Средства и способы связи для информационного обмена между компонентами

ПО работает в информационно-коммуникационной среде интернет/интранет.

Для информационного обмена между компонентами используются следующие подходы:

* взаимодействие посредством внутренних программных интерфейсов (API) в онлайн-режиме;
* иными средствами.

Взаимодействие осуществляется по стандартизированным протоколам.

Информационное взаимодействие на сетевом уровне строится с использованием протоколов стека TCP/IP.

Обмен данными ведется в формате JSON.

JSON (англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

* набор пар ключ: значение. Возможны реализации как объект, запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка (регистр зависимая: имена с буквами в разных регистрах считаются разными), значением — любая форма;
* упорядоченный набор значений. Возможны реализации как массив, вектор, список или последовательность, т. е. универсальные структуры данных: как правило, любой современный язык программирования поддерживает их в той или иной форме. Данные структуры легли в основу JSON, и он используется для обмена данными между различными языками программирования.

В качестве значений в JSON могут быть использованы:

* объект —неупорядоченное множество пар ключ: значение, заключённое в фигурные скобки «{}». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми;
* массив (одномерный) —упорядоченное множество значений. Массив заключается в квадратные скобки «[]». Значения разделяются запятыми;
* число;
* литералы true, false и null;
* строка — упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты \', \", \\, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

Информационный обмен между ПО и клиентами осуществляется по протоколу HTTP(S).

Удаленное администрирование и обслуживание базового программного обеспечения ПО выполняется с использованием защищенных протоколов (SSH или аналогичных).

## Связи ПО с другими программами

Информационное взаимодействие на сетевом уровне строится с использованием протоколов стека TCP/IP.

Описать взаимодействие с:

* СУДИР
* Шлюз ЕЦХД
* Детектор ТС
* Внешний клиент АПИ

# Используемые технические средства

Технические средства (далее – ТС), обеспечивающие функционирование ПО, предоставляются Заказчиком и размещены на базе вычислительных ресурсов Заказчика.

Программное обеспечение должно корректно функционировать и выполнять показатели назначения на серверных электронно-вычислительных машинах со следующими минимальными характеристиками:

– процессор (CPU): не менее 24 ядер;

– оперативная память (RAM): не менее 64 ГБ;

– накопитель на магнитном диске либо твердотельный: не менее 6 ТБ полезной емкости;

– архитектура процессора и операционной системы: 64 разряда,

Программное обеспечение функционирует под управлением операционной системы М ОС.

АРМ пользователя должен размещаться на устройствах, обеспечивающих его индивидуальное использование авторизуемым пользователем (в рамках обеспечения безопасности в части доступа к информационной системе и ее данным), а также поддержку браузеров Chrome 85+.

В качестве периферийных ТС рекомендуется использование периферийных устройств сбора, подготовки и отображения данных:

– стандартная клавиатура и манипулятор типа «мышь»;

– монитор LCD с разрешением не менее 1440х900 и диагональю не менее 17”.

Для функционирования ПО на автоматизированных рабочих местах (далее – АРМ) пользователей рекомендуется использование ТС с характеристиками не ниже указанных в таблице «Технические характеристики АРМ пользователя» (таблица 6).

Таблица 6 – Технические характеристики АРМ пользователя

| ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ | ЗНАЧЕНИЕ, НЕ МЕНЕЕ |
| --- | --- |
| Системный блок: | |
| Частота процессора, ГГц | 3 |
| Тип оперативной памяти | «DDR 4» |
| Объем оперативной памяти, Гбайт | 8 |
| Частота шины оперативной памяти, МГц | 2666 |
| Тип запоминающего устройства | «HDD» или «SDD» |
| Объем запоминающего устройства, Гбайт | 100 |
| Мощность блока питания, Вт | 400 |

# Вызов и загрузка

## Способ вызова программы с соответствующего носителя данных

Пользователь может обращаться к программе, работая как на ПЭВМ под управлением Linux, так и на ПЭВМ под управлением Windows. Программа вызывается пользователем через браузер обращением по ее адресу посредством HTTP-запросов с аутентификацией.

Подготовка программы к работе и ее настройка администратором описаны в документе «Руководство администратора».

## Входные точки в программу

Адрес доступа сообщается пользователям администратором системы.

Пользователи ПО получают доступ к графическому интерфейсу ПО при обращении из браузера по Адресу доступа на HTTPS URL ПО.

Клиенты API ПО получают доступ к REST API по Адресу доступа на URL с соответствующим методом REST и параметрами запроса.

# Входные данные

Программа поддерживает графический интерфейс работы с пользователем, поэтому все произведенные пользователем воздействия на программу можно рассматривать как входные данные в нее (рисунок 9, поток 3). Описание работы с программой через графический интерфейс приведено в документах:

«Руководство пользователя»;

«Руководство администратора».

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Автоматически созданное описание

Рисунок 9 – Схема информационных потоков

Описание информационных потоков представлено ниже (таблица 7).

Таблица 7 – Описание информационных потоков

| № ПОТОКА НА СХЕМЕ | ПЕРЕДАВАЕМЫЕ ДАННЫЕ |
| --- | --- |
| 1 | Отправка в СУДИР запроса на авторизацию |
| 2 | Получение из СУДИР JWT токена |
| 3 | Сохранение настройки заданий анализа парковочных мест, через интерфейс пользователя. |
| 4 | Получение сохраненных настроек, отображение данных камер, результатов анализа парковок. |
| 5 | Отправка изображения на детектор |
| 6 | Получение от детектора выявленных по изображению данных. |
| 7 | Запрос на получение свойств и изображений с камер. |
| 8 | Получение изображений с камер, свойств камер. |
| 9 | Запрос данных по занятости парковок. |
| 10 | Ответ со списком парковок и их статусом занятости. |

Описать параметры запросов, когда будет реализация API по координатам и УНОМ

# Выходные данные

Программа поддерживает графический интерфейс работы с пользователем, поэтому все высвеченные для пользователя формы и их информационное наполнение можно рассматривать как выходные данные программы. Описание работы с программой через графический интерфейс приведено в документах:

«Руководство пользователя»;

«Руководство администратора».

Описать объекты ответов, когда будет реализация API по координатам и УНОМ

### Структура данных

Данные содержат:

### Формат данных

Данные представляют набор значений в формате значений полей SQL-таблицы.

### Описание данных

Данные представляют количественную оценку

# ПЕРЕЧЕНЬ ТЕРМИНОВ

Термины и их определения представлены ниже (таблица 8).

Таблица 8 – Термины и их определения

|  |  |
| --- | --- |
| ТЕРМИН | ОПРЕДЕЛЕНИЕ |
| [Модуль распознавания регистрационных знаков и определения характеристик транспортных средств](https://confluence.amtech.ru/pages/viewpage.action?pageId=1066144072) | Программное обеспечения, предназначенное для распознавания свойств ТС и получения bbox ТС на исходном изображении |
| Парковочное место |  |
| Парковочное пространство |  |
| Шлюз ЕЦХД |  |
| СУДИР |  |
| bbox |  |
| HTTP | Протокол прикладного уровня передачи данных, в соответствии с [RFC 2616] |
| JSON | (JavaScript Object Notation) –текстовый формат обмена данными, в соответствии с [RFC 8259] |

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Сокращения и их расшифровки представлены ниже (таблица 9).

Таблица 9 – Сокращения и их расшифровки

| СОКРАЩЕНИЕ | РАСШИФРОВКА |
| --- | --- |
| АРМ | Автоматизированное рабочее место |
| БД | База данных |
| ГОСТ | Государственный стандарт |
| ОС | Операционная система |
| ПО | Программное обеспечение |
| СУБД | Система управления базами данных |
| ТС | Технические средства |
| API | (Application Programming Interface) — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой |
| DDR | (Double Data Rate) – динамическая память с удвоенной скоростью передачи |
| HDD | (Hard Disk Drive) – накопитель на жестком диске |
| HTTP | (HyperText Transfer Protocol) — протокол прикладного уровня передачи данных. |
| IP | (Internet Protocol) – протокол Интернета |
| SQL | (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных |
| SSD | (Solid-State Drive) – компьютерное энергонезависимое немеханическое запоминающее устройство на основе микросхем памяти |
| TCP | (Transmission Control Protocol) – один из основных протоколов передачи данных Интернета, предназначенный для управления передачей данных |
| TCP/IP | (Transmission Control Protocol/Internet Protocol) – сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет |
| XML | (eXtensible Markup Language) – расширяемый язык разметки |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Лист регистрации изменений** | | | | | | | | | |
| Изм | Номера листов (страниц) | | | | Всего листов (страниц) в документе | №  документа | Входящий № сопроводит. документа и дата | Подп. | Дата |
| Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |