|  |  |
| --- | --- |
|  | **УТВЕРЖДАЮ**  Генеральный директор  ЗАО «АВК-Коммьюникейшнз»  \_\_\_\_\_\_\_\_\_\_\_\_\_ (Быченков В.М.)  «25» февраля 2021г.  М.П. |

**Модуль АСР «Атлант»**

**ПЛАТФОРМА ЦИФРОВИЗАЦИИ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ «АТЛАНТ»**

**Описание платформы цифровизации производственных процессов**

**Листов 74**

2021

Оглавление

[1. Общие положения 5](#_Toc119877467)

[1.1. Наименование системы 5](#_Toc119877468)

[1.2. Цели, назначение, основные задачи и область использования системы 5](#_Toc119877469)

[1.2.1. Назначение АС 5](#_Toc119877470)

[1.2.2. Цель создания АС 5](#_Toc119877471)

[1.3. Техническое обеспечение, необходимое для функционирования программы 6](#_Toc119877472)

[1.4. Программное обеспечение, необходимое для функционирования программы 6](#_Toc119877473)

[1.5. Языки программирования и инструменты, с использованием которых написана программа 7](#_Toc119877474)

[2. Логическая и функциональная архитектура системы 8](#_Toc119877475)

[2.1. Сторонние технологические компоненты 8](#_Toc119877476)

[2.2. Базовая компонентная архитектура 9](#_Toc119877477)

[2.2.1. Модуль Интеграционный адаптер 10](#_Toc119877478)

[2.2.2. Модуль управления цифровыми моделями процессов и бизнес-сущностей 11](#_Toc119877479)

[2.2.3. Модуль исполнения процессов. 13](#_Toc119877480)

[2.2.4. Модуль управления задачами 14](#_Toc119877481)

[2.2.5. Модуль распределенного хранилища данных ПЦПП «Атлант» 14](#_Toc119877482)

[2.2.6. Сервисная шина 15](#_Toc119877483)

[2.2.7. Модуль управления правами доступа (ACM-модуль) 16](#_Toc119877484)

[2.2.8. Модуль взаимодействия с пользователем (поддержки веб-интерфейса) 17](#_Toc119877485)

[2.2.9. Модуль анализа журналов системы 22](#_Toc119877486)

[3. Обеспечение информационной безопасности 23](#_Toc119877487)

[4. Структура системы 25](#_Toc119877488)

[4.1. Решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости 27](#_Toc119877489)

[4.2. Решения по режимам функционирования, диагностированию работы системы 28](#_Toc119877490)

[4.2.1. Режимы работы системы 28](#_Toc119877491)

[4.2.2. Протоколирование работы платформы 29](#_Toc119877492)

[5. Состав функций и задач, реализуемых системой 31](#_Toc119877493)

[5.1. Представление в цифровом виде моделей производственных и деловых процессов 31](#_Toc119877494)

[5.2. Загрузка цифровых моделей в систему, менеджмент моделей. 32](#_Toc119877495)

[5.3. Формирование экземпляров атомарных задач и/или уведомление о наступлении того или иного события. 34](#_Toc119877496)

[5.4. Исполнение (обработка) атомарной задачи соответствующим участником единого информационного пространства. 36](#_Toc119877497)

[6. Описание информационной базы 38](#_Toc119877498)

[7. Пользовательский интерфейс 42](#_Toc119877499)

[7.1. Требования к рабочим местам пользователей ПЦПП «Атлант» 43](#_Toc119877500)

[8. Входные данные 43](#_Toc119877501)

[9. Выходные данные 44](#_Toc119877502)

[ПРИЛОЖЕНИЯ 46](#_Toc119877503)

[Приложение №1 46](#_Toc119877504)

[Приложение №2 47](#_Toc119877505)

[Приложение №3 50](#_Toc119877506)

[Приложение №4 52](#_Toc119877507)

# 

# Общие положения

## Наименование системы

Полное наименование разрабатываемой системы – Модуль АСР «Атлант» платформа цифровизации производственных процессов «Атлант».

Краткое наименование разрабатываемой системы – ПЦПП «Атлант», M-EDF Atlant.

## Цели, назначение, основные задачи и область использования системы

Разрабатываемая ПЦПП «Атлант» предназначена для решения задач интеграции и автоматизации сквозных производственных процессов. Решение данных задач обеспечит необходимую платформу для подключения и вовлечения в сквозные производственные процессы как различных автоматизированных систем (унаследованных и перспективных), так и имеющегося промышленного оборудования, что предоставит необходимую технологическую базу для поэтапной реализации концепции фабрики будущего.

### Назначение АС

К виду автоматизируемой деятельности относятся:

а) процессы сбора (получения), обработки и предоставления (отображения) информации посредством ПЦПП «Атлант», включая:

1) создание цифровых моделей бизнес-сущностей, используемых в деловых процессах организации;

2) создание цифровых моделей деловых процессов и сквозных производственных процессов;

3) создание (в том числе путём получения из интегрируемых систем), редактирование и прочие виды обработки информации;

4) организация хранения информации;

5) обмен информацией между интегрируемыми информационными системами и оборудованием;

6) формирование отчетов.

б) технические, методические и организационные мероприятия по сопровождению ПЦПП «Атлант».

### Цель создания АС

Основной целью создания ПЦПП «Атлант» является решение задач интеграции гетерогенных информационных систем и оборудования в ходе реализации технологических и бизнес-процессов.

## Техническое обеспечение, необходимое для функционирования программы

Минимальные требования к аппаратному обеспечению

* CPU – 1 Core;
* RAM – 3 Gb;
* HDD – 100 Gb mid-range.

Минимальные требования могут меняться в зависимости от используемой операционной системы.

## Программное обеспечение, необходимое для функционирования программы

Поддерживаются следующие операционные системы.

Windows Client:

* Windows 10 x86 (32-bit);
* Windows 10 x64 (64-bit);
* Windows 8.x x86 (32-bit);
* Windows 8.x x64 (64-bit);
* Windows 7 x86 (32-bit);
* Windows 7 x64 (64-bit).

Windows Server:

* Windows Server 2016 x64 (64-bit);
* Windows Server 2012 R2 x64 (64-bit);
* Windows Server 2012 x64 (64-bit);
* Windows Server 2008 R2 SP1 x64 (64-bit).

Linux:

* Debian 11;
* Ubuntu Linux x86 (32-bit) 16.04;
* Ubuntu Linux x64 (64-bit) 16.04.

В качестве СУБД, на которой развернута база данных ПЦПП «Атлант, может использоваться СУБД PostgreSQL (версии не ниже 9.5).

В качестве источников данных из интегрируемых систем поддерживаются следующие СУБД:

* PostgreSQL;
* Microsoft SQL Server 2008, 2012;
* Oracle Database.

Потенциально возможно использовать в качестве источников данных любые СУБД, поддерживающие ANSI SQL.

Работа клиентской части (веб-интерфейс) поддерживается в следующих браузерах с включённой поддержкой сценариев JavaScript:

* Internet Explorer (версии 10 и выше);
* Microsoft Edge;
* Firefox;
* Google Chrome (версии 49 и выше);
* Apple Safari (версии 10 и выше).

## Языки программирования и инструменты, с использованием которых написана программа

Для создания ПЦПП «Атлант» использовано следующее программное обеспечение (как проприетарное, так и под opensource-лицензиями):

1. Операционные системы: Linux x86\_64, пакеты операционной системы: java openjdk 1.8.0\_181 и выше.
2. Средства разработки: разработка выполнена на языках Java и JavaScript.
3. Шина для передачи данных (тиражируемое связующее программное обеспечение, ориентированное на обработку сообщений). При разработке использовано свободное ПО RabbitMQ версии 3.7.4 и выше – программный брокер сообщений на основе стандарта AMQP. Состоит из сервера, библиотек поддержки протоколов HTTP, XMPP и STOMP, клиентских библиотек AMQP для Java и .NET Framework и различных плагинов. Имеется реализация клиентов для доступа к RabbitMQ для ряда языков программирования (в том числе Perl, Python, Ruby, PHP). Поддерживается горизонтальное масштабирование для построения кластерных решений. Выпускается под лицензией MPL, что позволяет также использовать RabbitMQ как составляющий компонент ПЦПП «Атлант» в сочетании с проприетарными файлами. По желанию заказчика платформы возможна интеграция другой транспортной шины (реализуется как доработка системы на заказ).
4. СУБД: в качестве СУБД используется PostgreSQL версии 9.5, кроме того, могут использоваться СУБД Oracle версии 11 и выше или Microsoft SQL Server версии 2016 и выше. По желанию заказчика платформы возможно использование любой RDBMS, поддерживающей JDBC (реализуется как доработка системы на заказ), а также MongoDB.
5. В качестве веб-сервера используется Node.js версии 9.11.2 и выше – opensource кроссплатформенная среда выполнения для JavaScript, которая работает на серверах. Для разработки пользовательских интерфейсов используется Vue.js – JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов, а также Vuetify – библиотека компонентов графического интерфейса.

Для работы с БД используется язык запросов SQL в рамках стандарта ANSI SQL-92.

Для разработки пользовательских интерфейсов используются встроенные возможности Node.js и Vue.js, а также, в случае необходимости, языки SQL, Java 1.4 и выше, язык разметки гипертекста – HTML 3.2 и выше, Java Script 1.3 и выше.

Моделирование схемы данных выполняется в нотации UML Domain Classes (UМL 2.4.1). Моделирование бизнес-процессов осуществляется в нотации BPMN.

# Логическая и функциональная архитектура системы

## Сторонние технологические компоненты

В состав разрабатываемой системы включены следующие технологические компоненты:

– Сервер БД – представляет собой промышленную систему управления базами данных (СУБД). На сервере БД хранятся служебные и пользовательские данные платформы. Реализована система разграничений прав доступа на уровне схем. В качестве сервера БД возможно использовать:

* PostgreSQL (версии 9.6.2. и выше);
* Microsoft SQL Server (версии 2016 и выше);
* ORACLE (версии 11 и выше).

– Брокер сообщений – компонент, с помощью которого реализована шина сообщений и сервисная шина ПЦПП «Атлант». Сервисная шина отвечает за взаимодействие модулей в части обмена информацией и работает на основе брокера сообщений. В качестве брокера сообщений используется RabbitMQ версии 3.7.4 или выше.

– Опциональный модуль анализа журналов Системы использует набор следующих свободно распространяемых программных продуктов:

* Elasticsearch – поисковая система;
* Kibana – платформа аналитики и визуализации данных с открытым исходным кодом, используемая для работы с Elasticsearch;
* Logstash – система сбора данных с открытым исходным кодом с возможностями конвейерной обработки данных в реальном времени;
* Агенты сбора данных Collectd (сбор данных о состоянии сервера), Filebeat (сбор данных из журналов модулей, хранящихся на файловой системе серверов), загружающие информацию в БД Logstash.

– Средство администрирования баз данных – предлагается использовать нативные продукты эксплуатируемой СУБД или IDE сторонних разработчиков по выбору пользователя.

- Средства администрирования и разработки.

## Базовая компонентная архитектура

Архитектурно основными элементами платформы являются ядро и функциональные модули (см. Рисунок 1). При этом ядро системы само по себе также является набором базовых функциональных модулей (системных, не подлежащих доработке и изменению):

* Модуль управления моделями. Модуль отвечает за функционал связанный с хранением, управлением и предоставлением информации о моделях.
* Модуль исполнения процессов. Модуль отвечает за реализацию алгоритмов по формированию, управлению и хранению экземпляров элементов, определенных в нотации BPMN 2.0. Основным результатом работы модуля являются экземпляры атомарных задач (BPMN: A Task us a unit of work, the job to be performed) и (или) уведомления о наступлении того или иного события (BPMN: Event), которые передаются на обработку (на выполнение) в Модуль управления задачами.
* Модуль управления задачами. Модуль отвечает за выполнение переданных ему атомарных задач (и обработку событий) при помощи присоединенных (зарегистрированных) функциональных модулей.



Рисунок 1 – Базовый состав решения

Информация о базовых модулях ядра приведена далее.

Функциональные модули – зарегистрированные в ядре исполнители атомарных задач. Любой функциональный модуль состоит из интеграционного адаптера, который обеспечивает общую бизнес-логику работы с ядром и непосредственно части реализующей ту или иную прикладную функциональную возможность.

Структурно разрабатываемые компоненты системы скомпонованы в следующие java-модули:

core

core-adapter-auth

core-adapter-autocreate

core-adapter-core

core-adapter-entity

core-adapter-model

core-adapter-script

edf-adapter-restproxy

### Модуль Интеграционный адаптер

Интеграционный адаптер является типовым программным настраиваемым элементом и входит в состав функционального модуля. Имеет собственный унифицированный программный интерфейс и обеспечивает информационный обмен с ядром Системы через сервисную шину и с другими адаптерами напрямую. Возможен вариант коммутации с другими адаптерами с использованием вспомогательных средств типа отдельной ESB. Обеспечивается путем публикации программного интерфейса адаптера на ESB.

Модуль обеспечивает следующие функциональные возможности:

* + Настройку типов исполняемых атомарных задач, включая дополнительную информацию, характеризующую затраты на исполнение задачи;
  + Настройку типов сущностей, которые модуль может предоставлять (get);
  + Настройку типов сущностей, сохранение которых модуль может обеспечить (set);
  + Настройку способностей исполнять другие типовые действия (например, исполнять операцию авторизации);
  + Выполнять регистрацию в качестве функционального модуля в ядре Системы;
  + Принимать атомарные задачи на исполнение;
  + Передавать информацию об исполнении атомарной задачи;
  + Принимать и отвечать на запрос о текущем статусе функционального модуля;
  + Подписываться на события определенных типов;
  + Передавать управляющему ядру информацию о возникновении событий определенных типов.

### Модуль управления цифровыми моделями процессов и бизнес-сущностей

Один из базовых модулей ядра системы. Отвечает за функционал, связанный с хранением, управлением и предоставлением информации о моделях.

Основное назначение модуля – хранение, передача и обработка информации:

* о классах объектов, которыми оперирует система, методах и взаимосвязях между ними;
* о принципах взаимодействия классов, моделях их использования и событиях, в которых они могут быть задействованы;
* о закономерностях и ограничениях использования классов, моделей классов, моделей обработки классов.

Модуль обеспечивает следующие функциональные возможности:

* Ведение каталога типов атомарных задач;
* Импорт моделей;
* Хранение моделей;
* Проверка моделей на целостность и непротиворечивость;
* Ввод и вывод моделей из использования;
* Модификацию моделей по правилам эволюционного развития;
* Поддержку версионности моделей;
* Поддержку областей видимости моделей;
* Разделение моделей по Проектам;
* Поддержку частной настройки моделей под требования конкретного Проекта.

Данный модуль является функционально независимым. Выполнение его задач не требует взаимодействия с другими модулями. В рамках выполнения задач модуля управления моделями отсутствует связанность с другими модулями.

Модели используемых системой понятий представлены в виде диаграмм классов (RDF). Для описания модели обработки понятий используется нотация BPMN – система условных обозначений для моделирования бизнес-процессов.

Диаграмма классов представляет собой структурную диаграмму языка моделирования UML, демонстрирующую общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

Модулем обрабатываются следующие элементы диаграммы классов:

1. Связи между сущностями (классами):

* зависимость;
* ассоциация;
* агрегация;
* композиция.

1. Типы событий:

* старт (начальное событие);
* финиш (конечное событие);
* сигнал.

1. Шлюзы (развилки):

* исключающий;
* параллельный.

1. Типы данных (позволяют определить формат ввода и отображения данных класса). Перечень типов данных может расширяться. При проектировании платформы определены следующие типы данных:

* checkbox;
* date;
* file;
* geopoint;
* image;
* link;
* number;
* radiobutton;
* text;
* text\_html;
* textarea.

Описание используемых типов данных приведено в приложении №2. Часть java-кода, описывающего обрабатываемые элементы, приведена в приложении №2.

### Модуль исполнения процессов.

Модуль отвечает за реализацию алгоритмов по формированию, управлению и хранению экземпляров элементов, определенных в нотации BPMN 2.0. В результате исполнения алгоритмов, заложенных в нотацию BPMN 2.0, данный модуль производит (формирует) экземпляры атомарных задач (BPMN: A Task us a unit of work, the job to be performed) и (или) уведомления о наступлении того или иного события (BPMN: Event), которые передаются на обработку (на выполнение) в Модуль управления задачами для дальнейшей обработки. В ответ модуль получает информацию о выполнении того или иного экземпляра задачи и (или) наступлении того или иного события, обеспечивает дальнейшее выполнения алгоритмов.

Данный модуль является функционально зависимым. Для выполнения своих задач он обращается к модулю управления моделями и модулю исполнения задач.

Модуль обеспечивает следующие функциональные возможности:

* Исполнение процессов согласно правил нотации BPMN 2.0;
* Обеспечение взаимодействия с модулем управления задачами;
* Предоставление информации об экземплярах процессов.

### Модуль управления задачами

Отвечает за выполнение переданных ему атомарных задач (и обработку событий) при помощи присоединенных (зарегистрированных) функциональных модулей, управляет взаимодействием с адаптерами исполняющих модулей:

* Шины сообщений и сервисной шины платформы цифровизации производственных процессов (отвечает за взаимодействие модулей в части обмена информацией);
* Интеграционного адаптера платформы цифровизации производственных процессов (отвечает за взаимодействие между платформой цифровизации производственных процессов и другими участниками единого информационного пространства);

Модуль обеспечивает следующие функциональные возможности:

* Регистрация функциональных модулей (включая идентификацию и авторизацию исполняющих модулей);
* Ведение оперативного реестра функциональных модулей;
* Получение атомарных задач на исполнение;
* Выбор функционального модуля для выполнения атомарной задачи;
* Формирование задания для функционального модуля;
* Получение результатов выполнения задания функциональным модулем;
* Передача информации об исполнении атомарной задачи.

### Модуль распределенного хранилища данных ПЦПП «Атлант»

Один из дополнительных модулей центральной части системы. Отвечает за хранение системных и пользовательских данных и обеспечивает следующие функциональные возможности:

* Использование для хранения данных реляционных СУБД PostgreSQL, Microsoft SQL Server, Oracle;
* Автоматическое создание схемы базы данных при первичном старте модуля;
* Поддержание БД в консистентном состоянии;
* Исполнение всей логики управления данными в БД без необходимости реализации какой-либо логики на стороне СУБД.

Фактически модуль является простейшей мастер-системой для сущностей класса «Message».

Структура универсального хранилища данных приведена на Рисунке 2.



Рисунок 2 – Концептуальная схема использования универсального хранилища данных

### Сервисная шина

Основной функцией модуля сервисной шины (шины сообщений) является обеспечение взаимодействия ядра системы с функциональными модулями. Сервисная шина реализована посредством RabbitMQ – свободно распространяемого бесплатного программного брокера сообщений на основе стандарта AMQP, ориентированного на обработку сообщений.

Модуль является буфером, обеспечивающим гарантированную обработку большого количества запросов / ответов и (или) соединений между внешними исполняющими модулями и модулями ядра.

Модуль обеспечивает следующие функциональные возможности:

* Публикация необходимых программных интерфейсов для взаимодействия с внешними исполняющими модулями (web service (xml), REST (json));
* Необходимую трансформацию данных между внешним программным интерфейсом и интерфейсами вызова функций ядра;
* Обеспечение возможности подключения большого количества исполняющих модулей (например, при реализации концепции построения IIoT (Industrial Internet of Things)).
* Обеспечение поддержки высоконагруженных режимов;
* Обеспечение гарантированной доставки данных.

### Модуль управления правами доступа (ACM-модуль)

Модуль отвечает за разграничение функционала и производственных процессов между пользователями. Взаимодействует с модулем распределенного хранения данных и модулем поддержки веб-интерфейса. Модулем реализуется авторизация пользователя и его права доступа к различным рабочим столам. Возможна интеграция с внешними средствами аутентификации и авторизации (например, LDAP). В состав ACM-модуля входит адаптер для доступа к каталогу Active Directory.

Объекты управления правами доступа. Определяются следующие объекты управления правами доступа:

* Модель данных
  + Класс
  + Атрибут
  + Связь
* Модель процесса
  + Процесс
    - Активность
    - Событие
    - Шлюз
    - Поток
* Экземпляры

Определяются следующие типы прав:

* Создание (Create)
* Чтение (Read)
* Изменение (Update)
* Удаление (Delete)

С точки зрения управления правами доступа ПЦПП «Атлант» рассматривается как распределенная система Single Sign-On (SSO), где одно приложение (service provider) делегирует функцию аутентификации пользователей другому приложению (identity provider). Например, модуль веб-интерфейса (service provider) делегирует функцию аутентификации модулю управления правами доступа (identity provider). Для подобных систем целесообразным является построение аутентификации на основе токенов.

На основе данной концепции описано решение следующих задач:

* Процедура аутентификации;
* Процедура авторизации.

**Процедура аутентификации.** В процедуре аутентификации выделяют следующие элементы:

* ***субъект аутентификации*** – любая сущность (пользователь, система, процесс и т.п.) способная инициировать выполнение действий в системе;
* ***характеристика субъекта*** — отличительная черта субъекта, который будет проходить процедуру аутентификации;
* ***механизм аутентификации,*** то есть принцип работы;
* ***аутентификационный токен,*** то есть результат процедуры аутентификации. Включает в себя достоверные сведения о субъекте аутентификации, которые используются в дальнейшем для идентификации и авторизации

В общем случае процедура аутентификации заключается в анализе предоставляемых *характеристик объекта* при помощи *механизма аутентификации* с целью формирования *аутентификационного токена*.

Последующее (повторное) выполнение процедуры аутентификации для идентичных *характеристик объекта* приведет к недействительности выданного ранее *аутентификационного токена.*

**Процедура авторизации.** В процедуре авторизации выделяют следующие элементы:

* ***субъект авторизации*** – любая сущность (пользователь, система, процесс и т.п.) способная инициировать выполнение действий в Системе;
* ***право*** – характеризует действие, выполняемое субъектом по отношению к объекту;
* ***объект*** – сущность в Системе, контроль за выполнение действий с которой осуществляется процедурой авторизации.

В общем случае под авторизацией следует понимать:

* процедуру проверки у ***субъекта авторизации*** наличия ***права*** по отношению к ***объекту***;
* процедуру изменения ***прав*** ***субъекта авторизации*** по отношению к ***объекту***.

### Модуль взаимодействия с пользователем (поддержки веб-интерфейса)

Модуль взаимодействия с пользователем является одним из исполняющих (прикладных) модулей ПЦПП «Атлант», обеспечивает выполнение атомарных типов задач, требующих взаимодействия с пользователем, т.е. отвечает за взаимодействие между платформой цифровизации производственных процессов и пользователями посредством графического интерфейса. Создает формы для отображения и заполнения пользовательских данных. Реализует адаптивный веб-интерфейс пользователя с поддержкой виджетов различного назначения. Обеспечивает автоматическую адаптацию пользовательского интерфейса при изменении моделей предметной области. Таким образом, модуль веб-интерфейса формирует отображение автоматизированного рабочего места (АРМ) пользователя для визуального отображения доступной данному пользователю информации и функционала платформы.

Веб-интерфейс Системы позволяет Пользователю переключаться между рабочими областями, оптимизированными для разной деятельности, связанными единым контекстом выполняемых задач.

В качестве способа взаимодействия с пользователем могут использоваться:

* графический веб-интерфейс;
* мобильное приложение.

***Графический веб-интерфейс (модуль взаимодействия с пользователем посредством веба)*** представляет собой современный адаптивный веб-интерфейс с возможностью настройки рабочих столов (workspaces) на основе визуальных компонентов (widgets|виджетов). Адаптивность подразумевает, что веб-интерфейс обеспечивает:

* ясное визуальное отображение информации независимо от характеристик экрана и других ограничений пользовательского устройства;
* автоматическую адаптацию пользовательского интерфейса при изменении моделей единой предметной области;
* возможность настройки существующих и создания новых интерфейсов.

Концепция взаимодействия с пользователем предполагает возможность настройки для пользователя различных рабочих столов, ориентированных на разные типы активности. Например, рабочий стол для навигации по данным (по аналогии с новостным порталом), рабочий стол для выполнения задач ориентированный на процессы, рабочий стол для формирования отчетности и т.д.

Каждый рабочий стол является совокупностью взаимосвязанных виджетов.

Виджет – это визуальный компонент, специализированный под определенный вид деятельности.

Примеры виджетов:

* главное меню;
* строка поиска;
* список;
* таблица;
* компонент для отображения или ввода адреса, и т.д.

Пользователь имеет возможность переключения между рабочими столами с сохранением контекста текущих действий (при наличии такой возможности).

***Мобильное приложение.*** В качестве мобильного приложения модуль взаимодействия с пользователем расширяет возможности адаптивного веб-интерфейса, включая возможность разработки, при необходимости, сложной интеграции с аппаратной составляющей смартфона/планшетного компьютера (Bluetooth, NFC, GPS, камера, датчики, подключаемые устройства и т.д.).

Модуль поддержки веб-интерфейса обеспечивает следующие функциональные возможности:

1. формирование современного адаптивного веб-интерфейса, в т.ч.:
   * автоматическая и контролируемая адаптация под изменение моделей предметной области;
   * возможность тонкой настройки интерфейсов;
   * подстраивается под разные размеры экрана;
2. создание и настройка шаблонов разметки рабочих областей;
3. конфигурирование виджетов и настройка алгоритмов взаимодействия между ними с использованием терминов и понятий моделей предметной области:
   * процессы / типы процессов;
   * атомарные задачи / типы атомарных задач;
   * сущности / классы данных;
   * атрибуты сущности / атрибуты класса данных;
   * связи между сущностями / связи между классами данных, и т.п.;
4. настройка оформления рабочих столов в т.ч. в зависимости от прав доступа, роли конкретного пользователя;
5. настройка индивидуального набора рабочих столов для пользователя;
6. обеспечение безопасности при передаче данных (ГОСТ-шифрование, использование HTTPS-протокола);
7. работа в режиме онлайн и оффлайн;
8. непрерывность работы на различных устройствах – при переходе с одного устройства на другое (например, из веб-интерфейса к мобильному приложению) в устройстве на котором пользователь продолжает работать, отражается вся информация, ранее сформированная через веб-интерфейс;
9. обеспечение взаимодействия с аппаратной составляющей смартфона/планшетного компьютера.

Таким образом, модуль веб-интерфейса обеспечивает:

1. Понятное и удобное для пользователя визуальное отображение информации независимо от характеристик экрана и других ограничений пользовательского устройства.
2. Формирование АРМ пользователя на основании моделей данных, процессов и сформированных экземпляров моделей, а именно:
   * визуальное отображение классов данных, сущностей, шагов моделей процессов, шагов процессов;
   * взаимодействие виджетов на основании связей моделей данных и моделей процессов.
3. Визуализацию только тех элементов моделей данных и/или моделей процессов, к которым пользователю предоставлен доступ.
4. Возможность использования рабочих столов, настроенных «по умолчанию».
5. Автоматическую настройку пользовательского интерфейса при изменении моделей (например, если в модель данных добавлен класс данных, связанный с отображаемым в пользовательском интерфейсе элементом, пользователю предоставляется возможность работать с данным элементом).
6. Возможность настройки бизнес-аналитиком пользовательского интерфейса с использованием каталога типов виджетов и правил, заложенных в модели. Бизнес-аналитик имеет возможность:
   * задавать область на размеченном рабочем столе, в которой будет располагаться необходимый виджет;
   * указывать элементы моделей, которые отображает виджет;
   * выбирать связи, на основании которых будут взаимодействовать выбранные виджеты, из доступных в соответствии с моделью данных (например, при выборе «Виджета №1» для отображения «Класса данных № 1», и «Виджета № 2» для отображения «Класса данных № 2» БА может указать, что данные виджеты взаимодействуют по связи: «Класс данных № 2» является частью «Класса данных № 1»);
   * задавать связь между рабочими столами интерфейса пользователя за счет указания связи между виджетами данных рабочих столов.

Модуль веб-интерфейса должен удовлетворять следующим требованиям:

1. Передаваемая между клиентом и веб-сервером информация должна быть защищена с применением средств криптографии. Для защиты данных должны использоваться криптографически стойкие алгоритмы и протоколы HTTPS (TLS 1.1).
2. Реализация веб-интерфейса («тонкого клиента») должна обеспечивать корректную работу следующих основных браузеров с включённой поддержкой сценариев JavaScript:
   * Internet Explorer (версии 10 и выше);
   * Google Chrome (версии 49 и выше);
   * Apple Safari (версии 10 и выше).
3. Взаимодействие модуля веб-интерфейса и ядра Системы должна осуществляться по API модулей ПЦПП «Атлант» (см. Приложение №4)
4. Хранение данных модуля должно осуществляться с использованием функциональных возможностей универсального модуля хранения данных.
5. Модуль должен реализовывать процедуру аутентификации.
6. Модуль должен иметь возможность настройки следующих способов аутентификации:
   * Аутентификация по паролю
   * Аутентификация при помощи электронной подписи
   * Многофакторная аутентификация с использованием кода подтверждения
7. Модуль должен предоставлять пользователю возможность доступа к разрешенной информации и действиям не выполняя процедуру аутентификации (поддерживать анонимный режим).
8. Модуль должен обеспечивать многопользовательскую работу со следующими характеристиками нагрузки
   * Количество обрабатываемых запросов – 10 000 запросов/день;
   * Общее количество зарегистрированных пользователей – 10 000;
   * Одновременное количество работающих пользователей – 200;
   * Время отображения главной страницы при максимальном числе одновременно работающих пользователей не более 2 секунд;
9. Модуль должен быть реализован с применением открытого (open source) программного обеспечения, обладающего необходимой лицензией для дальнейшего свободного коммерческого использования.
10. Для модуля должен быть определен класс сущности, атрибуты которого содержат всю информацию для его настройки. Конфигурирование модуля должно выполняться путем выполнения операции сохранения экземпляра сущности данного класса.
11. Модуль должен функционировать на оборудовании и операционных системах, указанных в соответствующих разделах документов «Верхнеуровневая архитектура ПЦПП «Атлант»»

### Модуль анализа журналов системы

Является опциональным и служит для предоставления пользователям удобного инструмента анализа журналов всех модулей системы, служащих для целей отладки процессов, наблюдения за корректностью работы Системы и исполняемых в ней процессов.

Результатом работы модуля является веб-страница, отображающая консоль администратора с информацией о серверах системы, а также интерфейс анализа журналов модулей системы с возможностью поиска и фильтрации.

# Обеспечение информационной безопасности

1. Идентификация и аутентификация.
   1. В модуле поддержки web интерфейса пользователи проходят аутентификацию и авторизацию. Авторизация Администратора в операционной системе серверов и СУБД осуществляется по отдельным аутентификационным данным настроенным непосредственно в ОС и СУБД.
   2. В системе отсутствуют групповые неперсонифицированные учетные записи, имеющие доступ к функциональности Системы. Учетные записи администраторов Системы не имеют доступа к функциональности Системы, предоставляемой посредством модуля поддержки web интерфейса.
   3. Любое действие пользователя в Системе возможно только после успешной аутентификации и авторизации. Для предоставления доступа к функциональности, Система определяет наличие разрешения на данный функционал по правам пользователя. Права пользователя определяются при авторизации в Системе.
   4. Для аутентификации используются стандартные и криптостойкие протоколы аутентификации пользователей. Аутентификация администратора осуществляется средствами ОС и СУБД. Канал связи клиента с сервером осуществляется по HTTPS протоколу.
   5. Системой используются идентификаторы пользователей, не указывающие на уровень привилегий пользователя.
   6. Пароли пользователей в Системе не хранятся.
   7. По умолчанию, в системе не реализуется противодействие атакам, направленным на подбор пароля.
2. Управление доступом (авторизация).
   1. Доступ к функциональности Системы осуществляется в зависимости от прав пользователя. Доступ Администраторов к Системе регулируется средствами ОС и СУБД.
   2. Управление учетными записями и правами пользователей осуществляется непосредственно в системе (при авторизации посредством протокола Kerberos, управление осуществляется на уровне Active Directory).
3. Протоколирование и аудит.
   1. В Системе реализованы функции протоколирования следующих событий безопасности: факты запуска процессов, изменения их состояния, исполнения атомарных задач, завершения процессов;
   2. Запуск функции аудита происходит автоматически при запуске Системы.
   3. Действия администратора протоколируются средствами ОС и СУБД.
   4. В Системе не поддерживается функциональность остановки аудита.
   5. Протоколируемые события сопоставляются с идентификаторами пользователей-инициаторов событий.
   6. Контроль доступа к журналам Системы, реализован средствами ОС и СУБД.
   7. Журналы регистрации событий хранятся в Системе централизованно в модуле распределенного хранилища данных. Протоколы работы модулей хранятся в журналах соответствующих модулей.
   8. Все данные в журналах регистрации событий хранятся в Системе не менее 12 месяцев.
4. Криптографическая поддержка.
   1. Информация, хранящаяся в модуле распределенного хранилища данных, защищена средствами СУБД.
   2. Для защиты передаваемых данных между Системой и пользователем используются криптографически стойкие алгоритмы и протоколы HTTPS (TLS 1.1).
5. Управление информационными потоками.
   1. В Системе используются стандартизированные протоколы и средства взаимодействия модулей Системы и внешних систем: HTTPS (TLS 1.2), HTTP, SSH, LDAP.
   2. Система не содержит публично доступных серверов. Все сервера Системы находятся внутри контура безопасности Покупателя.
6. Обеспечение отказоустойчивости и восстановления после сбоев.
   1. Для обеспечения возможности восстановления Системы после сбоев, связанных с потерей данных из-за аппаратных сбоев серверов Системы, необходимо периодическое сохранение резервных копий модулей системы и баз данных системы.
   2. В качестве холодного резерва используется среда Prodlike.
7. Система не содержит открытых API интерфейсов для интеграции в другие внешние системы.

# Структура системы

Система поддерживает размещение на виртуальных серверах. Схема развертывания зависит от конкретного решения. В минимальной конфигурации это может быть одна виртуальная машина с ОС Linuх и установленными СУБД PostgreSQL и VM Java 8, имеющая не менее 4 CPU, 8Gb RAM, 100Gb HDD.

Модули системы обладают слабой связанностью и для обеспечения большей производительности могут быть разнесены. В том числе допускаются варианты горизонтального масштабирования производительности путем дублирования функциональных модулей и построения иерархической структуры из нескольких ядер.

Стандартная конфигурация предполагает размещение в защищенном контуре сети всех модулей системы. На границе между модулем поддержи веб-интерфейса и пользователем устанавливается веб-сервер Nginx.

В таблице представлена структура продуктивной среды Production (см. пункт 3.7 текущего документа) в виде перечня и описания виртуальных машин, реализующих функциональность платформы. Соответствующая схема может быть реализована на различных физических машинах либо на сочетании физических и виртуальных машин (используя, например, Mikrotik RouterOS, Mikrotik CHR и т.п.).

Задвоение машин, содержащих те или иные компоненты, используется для обеспечения непрерывной работы платформы, возможности проведения ряда технических работ «на горячую» и, как следствие, обеспечения необходимого уровня SLA.

При составлении подобной схемы необходимо добавить в неё информацию о хостах/IP-адресах машин и параметрах доступа к ним (логин/пароль администратора машины).

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Номер виртуальной машины** | **Компонента** | **Краткое описание** |
| 1 | VM-1.1 | Например, Mikrotik RouterOS | Виртуальный маршрутизатор, обслуживает внутреннюю сеть |
| 2 | VM-1.2 |
| 3 | VM-2.1 | web\_adapter  nginx | Развернут проксирующий(proxy) сервер nginx, а также веб-модуль |
| 4 | VM-2.2 |
| 5 | VM-3.1 | core  core\_adapter  model\_adapter  auth\_adapter  entity\_adapter  script\_adapter  ESB (Rabbit MQ)  http rest api adapter | Развернуты структурные модули приложения:  core  core\_adapter  model\_adapter  auth\_adapter  entity\_adapter  script\_adapter  ESB (Rabbit MQ)  http rest api adapter |
| 6 | VM-3.2 |
| 7 | VM-4.1 | file\_adapter  file server | Развернуты структурные модули приложения:  file\_adapter  file server |
| 8 | VM-4.2 |
| 9 | VM-5.1 | PostgreSQL  Primary+StandBy | Развернута база данных, хранящая данные приложения |
| 10 | VM-5.2 |
| 11 | VM-6 | Backup | Развернутa машина, хранящая бэкапы базы данных. |
| 12 | VM-7 | Infra (CI/CD) | Развернутa машина, хранящая исходный код |
| 13 | VM-8 | Monitoring | Развернутa машина, предназначенная для мониторинга |
| 14 | VM-9 | Openvpn | Развернутa машина, осуществляющая доступ к сети компании |

## Решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости

Следует различать внешние и смежные системы. Смежные системы принадлежат пользователю ПЦПП «Атлант» и находятся в рамках его информационного контура. Внешние системы – не принадлежат (например, системы партнера, контрагента, ведомства).

С точки зрения ПЦПП «Атлант» взаимодействие с любой сторонней системой происходит однотипно. Различия между внешними и смежными системами не делается. Взаимосвязь с внешними системами является базовой функциональностью ПЦПП «Атлант».

За связь со сторонними системами отвечают интеграционные адаптеры. Интеграционный адаптер реализует взаимодействие между ПЦПП «Атлант» и другими участниками единого информационного пространства. Адаптер является типовым программным настраиваемым элементом. Имеет собственный унифицированный программный интерфейс и обеспечивает информационный обмен с управляющим ядром через сервисную шину и с другими адаптерами напрямую. Возможен вариант коммутации с другими адаптерами с использованием вспомогательных средств типа отдельной ESB (обеспечивается путем публикации программного интерфейса адаптера на ESB).

Общая схема настройки адаптера:

* получение и настройка учетной записи, под которой адаптер регистрируется в ядре ЕИС;
* настройка типов атомарных задач, которые может исполнять прикладной модуль (в состав которого входит данный адаптер)
* настройка классов данных сущностей, по отношению к которым прикладной модуль может выступать системой, отвечающей за создание, предоставление или хранение.

Пользовательская последовательность действий, реализующая процесс интеграции с внешней системой, может быть представлена следующим образом:

|  |  |
| --- | --- |
| **Действие** | **Функция/Роль** |
| 1. Создать пользовательскую модель | Бизнес-аналитик |
| * 1. Построить модель |  |
| * 1. Прописать пользовательские требования к содержанию и формату полей |  |
| * 1. Указать, как и для чего будут использоваться интеграционные данные |  |
| 1. Создать модель источника данных | Эксперт предметной области, проектировщик БД |
| * 1. Выделить классы данных и их атрибуты |  |
| * 1. Выстроить связи между классами |  |
| * 1. Алгоритм формирования классов и заполнения атрибутов в пользовательской модели из модели источника |  |
| 1. Настроить интеграционный адаптер с внешней системой | Системный аналитик |
| 1. Создать модели процесса импорта данных из внешней системы | Системный аналитик |
| * 1. Получить данные из внешней системы при помощи адаптера |  |
| * 1. Трансформировать данные внешней системы в данные для пользовательской модели |  |
| 1. Написать скрипт трансформации модели источника данных в пользовательскую модель (см. пункт 4.2) | Системный аналитик, Программист |
| 1. Согласовать и настроить регламент работы процессов импорта (см. пункт 4) | Бизнес-аналитик |

## Решения по режимам функционирования, диагностированию работы системы

### Режимы работы системы

Предполагается работа системы в одном из следующих режимов работы:

– Основной режим (все подсистемы выполняют свои функции, доступна работа авторизованных пользователей).

– Профилактический режим (недоступна работа авторизованных пользователей). В данный режим работы система переводится администратором в следующих случаях: возникновение необходимости модернизации аппаратно-программного комплекса; возникновение необходимости проведения технического обслуживания.

– Аварийный режим (одна или все подсистемы не выполняют своих функций). В данный режим система переходит в следующих случаях: выход из строя аппаратно-программного комплекса, вызванный выходом из строя элементов аппаратной или программной базы; выход из строя сети передачи данных и другие аварийные ситуации.

В основном режиме функционирования система обеспечивает:

– работу пользователей в режиме – 24 часа в день, 7 дней в неделю (24х7);

– выполнение своих функций – управление цифровыми моделями сущностей и процессов; сбор, обработку, загрузку и хранение данных; выполнение технологических и бизнес-процессов.

В профилактическом режиме система обеспечивает возможность проведения следующих работ:

– техническое обслуживание;

– модернизация составляющих аппаратно-программного комплекса.

В аварийном режиме работы система не выполняет свои функции. Допускается только устранение аварийных ситуаций с дальнейшим переводом системы в иной режим работы.

Общее время нахождения системы в профилактическом режиме не должно превышать 0,3% от общего времени работы системы (2,16 часа в месяц).

Общее время нахождения системы в аварийном режиме не должно превышать 0,4% от общего времени работы системы (2,88 часа в месяц).

С целью обеспечения высокой надежности функционирования системы необходимо проводить регулярное диагностирование её состояния (как системы в целом, так и отдельных её компонентов).

В ПЦПП «Атлант» не предусмотрено встроенное средство мониторинга работоспособности и производительности. В качестве средств мониторинга могут быть использованы сторонние системы, такие как Zabbix, BMC Patrol, Zenoss Core, IBM Tivoli, HP OpenView и т.п.

### Протоколирование работы платформы

В Системе происходит протоколирование действий пользователей, ведущих к добавлению или изменению данных в Системе, с привязкой действий к аутентификационным данным пользователей. Все протоколирование ведется централизованно в модуле исполнения процессов.

Протоколированию подлежат:

• Операции по созданию процессов;

• Операции по изменению статуса (выполнению) задач;

• Факты наступления событий;

• Операции по завершению обработки процессов.

# Состав функций и задач, реализуемых системой

Концепция автоматизации сквозных производственных процессов на основе цифровых моделей построена на использовании нотации BPMN и реализации функциональных алгоритмов BPM-машины. Соответственно, ПЦПП «Атлант» должна обеспечивать выполнение функций:

* представление в цифровом виде моделей производственных и деловых процессов;
* загрузка цифровых моделей процессов в систему, менеджмент моделей;
* формирование экземпляров атомарных задач и/или уведомление о наступлении того или иного события;
* исполнение (обработка) атомарной задачи соответствующим участником единого информационного пространства.

## Представление в цифровом виде моделей производственных и деловых процессов

Данный комплекс задач реализуется путём создания цифровых моделей бизнес-сущностей и бизнес-процессов.

1. **Моделирование данных единого информационного пространства.**

Система позволяет реализовать создание логической и физической модели данных, а также проектирование классов, отображающих бизнес-сущности, и связей между ними.

В разрезе данной документации реализация данного функционала осуществляется с помощью Sparx Systems Enterprise Architect версии 12 или выше (см. п.3.1.1.).

Формирование описания данных осуществляется в формате UML Domain Classes или другом аналогичном промышленном формате. Описываются классы данных, атрибуты классов, связи между классами, значения атрибутов по умолчанию, ограничения значений атрибутов, форматы данных и тому подобное.

1. **Моделирование сквозных процессов.**

Система позволяет реализовать создание цифровой модели бизнес-процессов. В разрезе данной документации реализация данного функционала осуществляется с помощью Sparx Systems Enterprise Architect версии 12 или выше (см. п.3.1.1.).

Для активностей (шагов) процессов, связанных с обработкой данных, формируются правила обработки (нормализация, проверка непротиворечивости, целостности и т.п.)

Для активностей (шагов) процессов, связанных с обменом данными с внешними системами (интеграционный обмен) формируются требования к адаптерам доступа к внешним информационным системам (SQL, REST, SOAP, RFC, custom API).

Описание процессов производится в формате BPMN 2.0. Описываются роли участников процессов, состав активностей (шагов) процесса, типы активностей, входящие и исходящие данных для каждого шага, элементы управления движения по процессу (шлюзы, условия перехода, события).

|  |  |
| --- | --- |
| **Подзадача** | **Действие** |
| Создание схемы данных (цифровой модели данных, модули классов) | В Модели классов описываются объекты, их свойства и отношения между этими объектами.  Модель классов строится на базе RDF-диаграммы. RDF-диаграмма описывает классы (объекты), их атрибуты (свойства) и связи (отношения) между этими классами.  В разрезе данной документации для создания RDF-диаграмм используется редактор Sparx Enterprise Architect версии 12.1 или выше. Модель выгружается в файл в формате xml. |
| Создание модели процесса | Модель процесса описывает течение процесса, последовательность действий процесса, результаты действий процесса, участников процесса.  Модель процесса строиться на базе Collaboration-диаграммы в рамках нотации BPMN 2.0.  Модель процесса опирается на модель данных. То есть для конкретной Collaboration-диаграммы всегда существует соответствующая конкретная RDF-диаграмма.  В разрезе данной документации для создания Collaboration-диаграмм используется редактор Sparx Enterprise Architect версии 12.1 или выше. Модель выгружается в файл в формате xml. |

## Загрузка цифровых моделей в систему, менеджмент моделей.

Загрузка в ПЦПП «Атлант» цифровых моделей схем данных и процессов реализуется с помощью Model Management Module (MMM) – Модуля управления моделями.

|  |  |
| --- | --- |
| **Подзадача** | **Действие** |
| Загрузка цифровой модели процесса | Реализуется с помощью Модуля управления моделями путём запуска процесса «МММ – Загрузка модели». Процесс загрузки модели позволяет:   * провести проверку загружаемой модели процесса; * загрузить модель в систему. |
| Редактирование цифровой модели процесса | В разрезе данной документации редактирование цифровой модели процесса производится с помощью редактора Sparx Enterprise Architect версии 12.1 или выше. Модель выгружается в файл в формате xml и повторно загружается в систему. Тем самым обеспечивается версионность развития процессов (экземпляры процесса, находящиеся в процессе выполнения, будут выполняться по старой модели процесса, а вновь запущенные – по новой). |
| Удаление цифровой модели процесса | Не предусмотрено |
| Загрузка цифровой модели схемы данных | Реализуется с помощью Модуля управления моделями путём запуска процесса «МММ – Загрузка модели». Процесс загрузки модели позволяет:   * провести проверку загружаемой модели схемы данных; * загрузить модель в систему. |
| Редактирование цифровой модели схемы данных | В разрезе данной документации редактирование цифровой модели схемы данных производится с помощью редактора Sparx Enterprise Architect версии 12.1 или выше. Модель выгружается в файл в формате xml и повторно загружается в систему. |
| Удаление цифровой модели схемы данных | Не предусмотрено |
| Просмотр цифровой модели процесса | Может быть настроен рабочий стол, отображающий перечень элементов процесса, их типов и связей между ними. |
| Просмотр цифровой модели данных | Может быть настроен рабочий стол, отображающий перечень классов и их атрибутов. |
| Извлечение цифровой модели процесса | При загрузке валидной цифровой модели процесса в системе сохраняется xml-файл модели, который может быть выгружен штатными средствами ПЦПП «Атлант». |
| Извлечение цифровой модели данных | При загрузке валидной цифровой модели данных в системе сохраняется xml-файл модели, который может быть выгружен штатными средствами ПЦПП «Атлант». |

## Формирование экземпляров атомарных задач и/или уведомление о наступлении того или иного события.

Согласно загруженной модели процесса при запуске данного процесса на выполнение осуществляется формирование экземпляров атомарных задач, которые выполняются ядром либо передаются функциональным модулям. Базовый модуль системы в основе своей работы использует модель данных. Модель данных (Model Storage Module, MSM) является системной, то есть она не доступна пользователям и не предполагает внесение изменений (только самим разработчиком). Действия, составляющие данную функцию, выполняются ядром системы, то есть без участия пользователя и незаметно для него (за исключением запуска процесса пользователем).

|  |  |
| --- | --- |
| **Подзадача** | **Действие** |
| Запуск процесса | Для работы процесса его необходимо запустить, при этом в системе создается экземпляр конкретной модели процесса. Согласно модели процесса, запуск процесса равнозначен выполнению стартового события процесса.  Одновременно в системе может быть запущенно несколько экземпляров как одной модели процесса, так и нескольких моделей.  Одновременно ядро может обрабатывать неограниченное количество процессов.  Ядро поддерживает следующие способы запуска процесса:   * из интерфейса (пользователь выбирает соответствующий процесс из перечня процессов и запускает его); * по расписанию. |
| Выполнение действия | Согласно модели процесса, разные действия могут исполняться разными ролями.  Порядок выполнения действий и роль исполнителя определяются ядром на основании модели процесса. Роль исполнителя каждого действия определяется тем, какие действия на каких дорожках расположены.  Разные типы действий исполняются разными модулями. Когда ход процесса достигает какого-либо действия, ядро на основании типа этого действия определяет исполнительный модуль для выполнения этого действия (см. 3.6.4). |
| Исполнение шлюза | Исполнение шлюза осуществляется ядром, независимо от его расположения внутри пула.  Исполнение шлюза начинается с определения его типа.  Если шлюз параллельный и разделяющий, то по ходу процесса сразу несколько действий передаются на исполнение.  Если шлюз параллельный и объединяющий, то после завершения всех параллельных потоков ядро запускает на выполнение следующий шаг.  Если шлюз эксклюзивный и разделяющий, то после проверки условия и выбора потока операций на исполнение передается только одно действие.  Если шлюз эксклюзивный и объединяющий, то ядро запускает на выполнение следующий шаг сразу после завершения одного из потоков, а все остальные потоки игнорируются. |
| Завершение процесса | Согласно модели процесса, завершение процесса равнозначно выполнению конечного события процесса.  До завершения процесс является активным. В момент завершения ядро фиксирует данные, экземпляр процесса попадает в архив и более не может быть изменен. |
| Заполнение информации | Ядро полностью логирует жизнедеятельность процесса от запуска до завершения. Информация хранится в соответствии с моделью данных и доступна для просмотра через веб-интерфейс.  Перечень сохраняемой информации приведён в приложении №3. |

## Исполнение (обработка) атомарной задачи соответствующим участником единого информационного пространства.

Выполнение атомарной задачи производится соответствующим участником единого информационного пространства. Ядро передаёт исполнение соответствующему модулю на основании типа этого действия. Система поддерживает следующие типы задач: пользовательский шаг (реализуется пользователем посредством взаимодействия с модулем веб-интерфейса), скриптовый шаг (реализуется модулем, выполняющим скрипты), системный шаг (реализуется ядром системы), сервисный шаг (реализуется внешними программными компонентами).

|  |  |
| --- | --- |
| **Подзадача** | **Действие** |
| Выполнение задачи модулем веб-интерфейса (пользовательская задача) | Пользовательская задача подразумевает активность человека, которому для выполнения задачи необходим некий рабочий интерфейс. С этой целью в системе предусмотрен специализированный веб-интерфейс.  Для исполнения пользовательской задачи ядро вызывает модуль веб-интерфейса. |
| Выполнение задачи модулем исполнения скриптов | Для исполнения скриптового шага ядро вызывает Модуль исполнения скриптов.  Скриптовый шаг подразумевает вызов (по имени скрипта) и выполнение заранее написанных и загруженных в систему скриптов. |
| Выполнение сервисной задачи | Сервисная задача подразумевает активность внешнего модуля (по сути, некоторой подключенной библиотеки). Для исполнения сервисной задачи ядро вызывает некоторую библиотеку, которая может выполнять некие действия либо самостоятельно, либо являясь интерфейсом к некоторой внешней системе (оборудованию, компоненту). В случае взаимодействия с внешней системой такой вариант используется в случае невозможности доступа к системе через интеграционный адаптер (стандартный механизм взаимодействия). |
| Выполнение системной задачи | Системная задача – это по сути подвид сервисной задачи: сервисная задача, выполняемая ядром. Возможные типы системных задач приведены в приложении №3 (см. Таблицу 2, описание возможных значений поля activityTypeId в информации о шаге) |

# Описание информационной базы

Создание базы данных и схемы производится на этапе инсталляции. Создание таблиц и последовательностей (sequence) выполняется скриптами автоматического развертывания Системы в соответствии с руководством администратора.

Наименование базы данных – edf. База данных edf содержит 4 схемы:

* core,
* entity\_adapter,
* auth,
* script.

1. Схема core предназначена для хранения информации о структуре и логике работы системы, а также декомпозированной информации о цифровых моделях данных и бизнес-процессов. Структура схемы core представлена на Рисунке 3.

Скрипты SQL, генерирующие объекты схемы (для СУБД Oracle) и отражающие атрибутивный состав таблиц схемы, представлены в приложении №2.

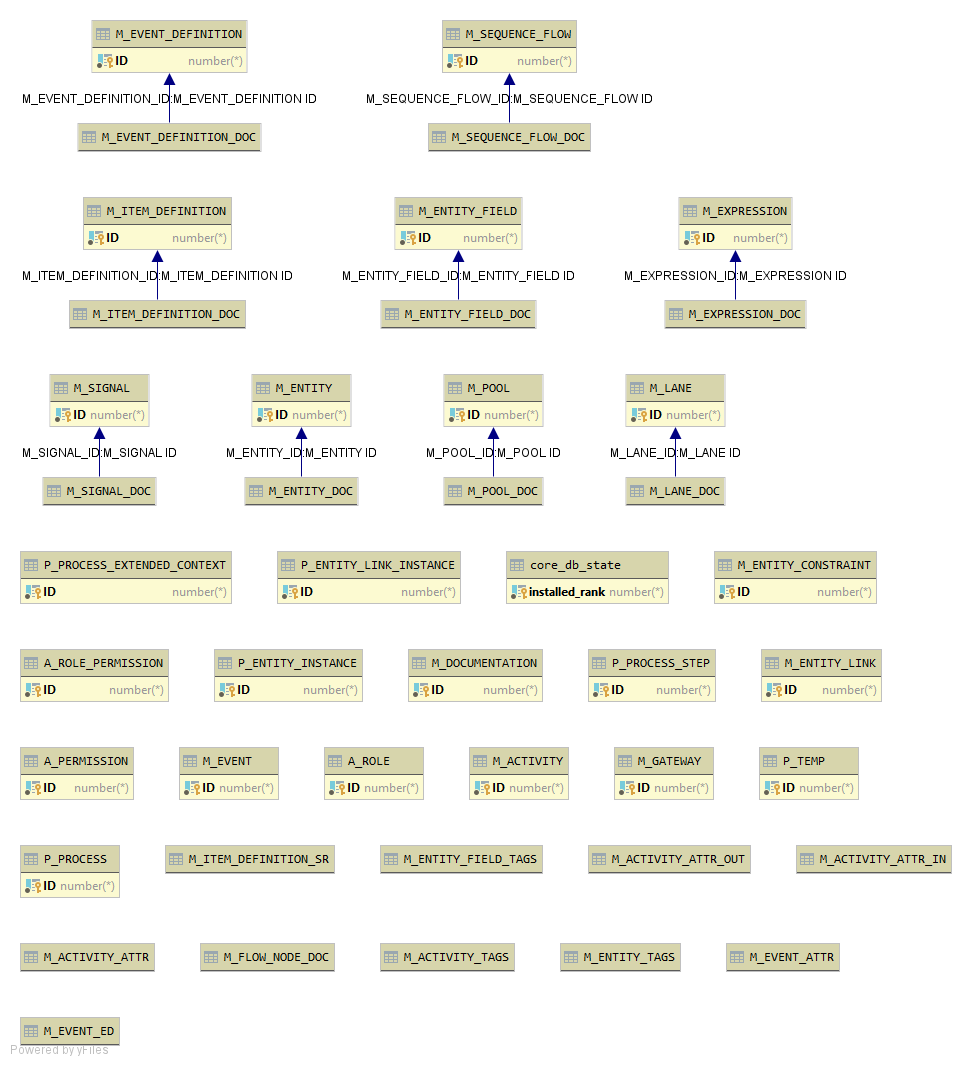


Рисунок 3 – схема core

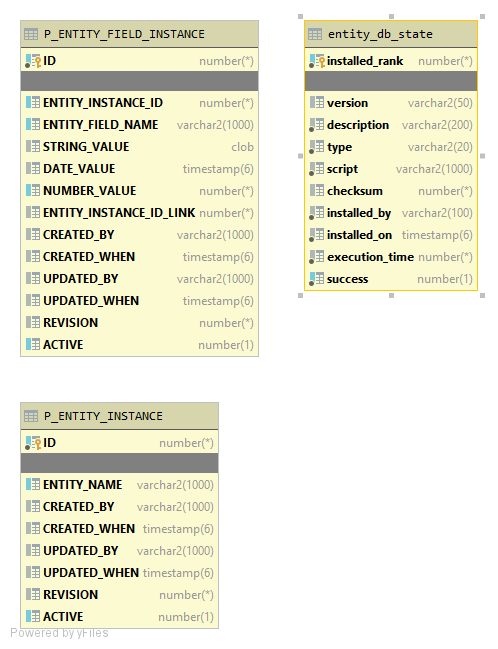
1. Схема entity\_adapter – служит для хранения пользовательской информации (загружаемой и преобразуемой в ходе выполнения технологических и бизнес-процессов пользователя). Данные хранятся в денормализованном виде в двух таблицах (см. Рисунок 4). Архитектура системы позволяет выполнить дальнейшую доработку для хранения данных в нормализованном виде либо подключения объектной БД. 

Рисунок 4 – схема entity\_adapter

1. Схема auth служит для работы адаптера модуля ACM, реализующего функционал AAA (Authentication, Authorization, Accounting). Структура схемы auth представлена на Рисунке 5.

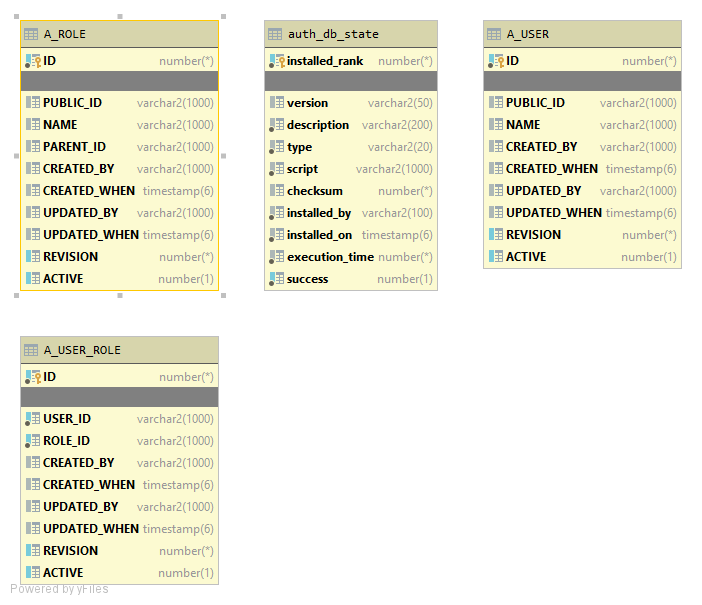


Рисунок 5 – схема auth

1. Структура схемы script представлена на Рисунке 6.

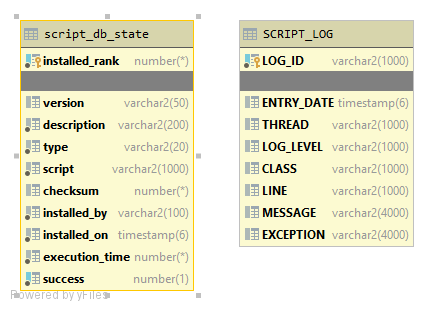


Рисунок 6 – схема script

При именовании таблиц БД соблюдаются следующие соглашения:

1. Префикс a\_ перед названием таблиц означает, что данные таблицы используются для авторизации пользователя в случае использования внутренней авторизации.

2. Префикс m\_ перед названием таблиц означает, что таблицы содержат логику работы ПЦПП «Атлант». Содержимое таблиц является частью кода системы и не подлежит изменению и/или настройке пользователями.

3. Префикс p\_ перед названием таблиц означает, что таблицы содержат информацию о созданных сущностях в ходе выполнения процессов.

# Пользовательский интерфейс

Доступ пользователей к системе осуществляется посредством браузеров Google Chrome, Microsoft Edge, Microsoft Internet Explorer, Mozilla Firefox по https-протоколу.

## Требования к рабочим местам пользователей ПЦПП «Атлант»

В разделе "Используемые технические средства" должны быть указаны типы электронных вычислительных машин и устройств, которые используются при работе программы.

Требования к рабочим местам пользователей – исполнителей бизнес-процессов:

Ноутбук/персональный компьютер:

* ОС – Windows7 или выше;
* CPU – не хуже Intel Core i5;
* RAM – не менее 4Gb;
* Браузер с поддержкой Java-скриптов – Internet Explorer (версии 10 и выше) или Google Chrome (версии 49 и выше).

Ноутбук/персональный компьютер под управлением ОС MacOS:

* ОС – MacOS 10 и выше;
* CPU - не хуже Intel Core i5;
* RAM – не менее 4 Gb;
* Браузер с поддержкой Java-скриптов – Apple Safari (версии 10 и выше), Google Chrome (версии 49 и выше).

Требования к рабочим местам пользователей – бизнес-аналитиков:

Ноутбук/персональный компьютер:

* ОС – Windows7 или выше;
* CPU – не хуже Intel Core i5;
* RAM – не менее 4Gb;
* любой SSH-клиент для консольного доступа к серверам Системы (например, Putty);
* Microsoft Word 2013 или выше.

# Входные данные

Приведение поступающей в систему информации к виду, пригодному для обработки с помощью ЭВМ, может происходить с данными, являющимися:

* данные, получаемые с оборудования и/или датчиков (пример: интерфейсы к SCADA-системам для снятия данных с промышленного оборудования, IoT и т.п.);
* информационные объекты, получаемые в результате анализа бизнес-процессов и загружаемые в систему (модели бизнес-процессов в формате BPMN 2.0, модели данных в формате UML 2.1.2 (XMI 2.1), скрипты на языках программирования SQL и groovy) для интерпретации ПЦПП «Атлант»;
* данные, получаемые от интегрируемых участников единого информационного пространства;
* запросы на данные, получаемые от интегрируемых участников единого информационного пространства.

Отличительной особенностью ПЦПП «Атлант» является возможность её настройки таким образом, чтобы исключить необходимость постоянной специальной подготовки данных.

Для обработки данных, получаемых от оборудования/датчиков, настраивается соответствующий адаптер. Данные передаются в ПЦПП «Атлант» в том виде, в котором их генерирует устройство.

Информационные объекты, загружаемые в систему (модели бизнес-процессов и модели данных), формируются в формате XMI 2.1 автоматически с помощью используемых для моделирования программных продуктов (например, Sparx Enterprise Architect).

Скрипты на языке groovy распознаются системой, скрипты на языке SQL требуют предварительной разметки, описанной в пользовательской документации.

Данные, получаемые от интегрируемых участников единого информационного пространства, могут быть представлены в любом виде. Требуется единоразово настроить соответствующий адаптер, а также создать в модели данных и бизнес-процесса нужную сущность.

Запросы на данные, получаемые от интегрируемых участников единого информационного пространства, требуют единоразовой настройки соответствующего интеграционного адаптера.

В случае, когда данные, получаемые от интегрируемых участников единого информационного пространства, требуют осуществления ручного ввода с использованием элементов веб-форм ввода данных, подготовка и предварительная валидация таких данных осуществляется персоналом компании, использующей ПЦПП «Атлант».

# Выходные данные

Выходные данные и управляющие команды, которые формирует и передает система:

* Информационные сообщения об успешной загрузке цифровой модели процесса и модели данных;
* Протоколы выполнения атомарной задачи (шага процесса);
* данные, передаваемые интегрируемым участникам единого информационного пространства;
* запросы на данные, передаваемые интегрируемым участникам единого информационного пространства.

# ПРИЛОЖЕНИЯ

## Приложение №1



Рисунок 8 – Архитектура ПЦПП «Атлант».

## Приложение №2

Используемые при моделировании процессов типы данных.

| **Тип данных** | **Ввод и вывод** | **Значение по умолчанию** | **Примечания** |
| --- | --- | --- | --- |
| checkbox | Ввод:   * активировать или деактивировать поле | * по умолчанию checkbox неактивен, то есть значение = False * можно указать значение = True, при котором checkbox по умолчанию будет активен | * отмеченный имеет значение True * неотмеченный имеет значение False |
| date | Ввод:   * ручной ввод * выбор из календаря   Вывод:   * формат по умолчанию → DD.MM.YYYY * другой формат задается через тэг format | Варианты значений:   * today\_current\_time → текущая дата и текущее время * today\_start\_day → * yesterday\_current\_time → вчерашняя дата и текущее время * yesterday\_start\_day → * yesterday\_end\_day → |  |
| file | Ввод:   * модальное окно выбор файла   Вывод:   * ссылка для скачивания файла |  | имя файла не поддерживает кириллицу |
| geopoint | Ввод:   * географические координаты |  |  |
| image | Ввод:   * модальное окно выбора файла   Вывод:   * картинка |  | просмотр картинки доступен только в виджете "карточка" |
| link | Ввод:   * URL   Вывод:   * гиперссылка |  |  |
| number | Ввод:   * число |  |  |
| radiobutton | Ввод:   * позволяет отметить один из вариантов * варианты задаются через тэг values * при наличии тэга select со значением True вместо radiobutton отображается DropDownList |  |  |
| text | Ввод:   * текст в одну строку |  |  |
| text\_html | Ввод:   * текст с применением html |  |  |
| textarea | Ввод:   * многострочный текст |  |  |

## Приложение №3

При исполнении процессов вся информация о процессе сохраняется в системе. Атрибуты и значения сохраняемой информации приведены ниже.

Таблица 1 – Сохраняемая информация о процессе.

| **Атрибут** | **Значение** | **Чем заполняется** |
| --- | --- | --- |
| createdBy | Кем создан | Имя пользователя |
| createdWhen | Когда создан | Дата и время создания |
| endTime | Когда завершен | Дата и время завершения |
| updatedWhen | Когда обновлен | Дата и время обновления |
| startTime | Когда запущен | Дата и время запуска |
| updatedBy | Кем обновлен | Имя пользователя |
| pk | Идентификатор | Уникальное целое число |
| poolPk | Идентификатор пула | Уникальное целое число |

Таблица 2 – Сохраняемая информация о процессе.

| **Атрибут** | **Значение** | **Чем заполняется** |
| --- | --- | --- |
| createdBy | Кем создан | Имя пользователя |
| createdWhen | Когда создан | Дата и время создания |
| endTime | Когда завершен | Дата и время завершения |
| updatedWhen | Когда обновлен | Дата и время обновления |
| startTime | Когда запущен | Дата и время запуска |
| updatedBy | Кем обновлен | Имя пользователя |
| pk | Идентификатор шага | Уникальное целое число |
| flowNodePk | Идентификатор типа шага | Уникальное целое число |
| status | Статус выполнения шага | SUCCESS – выполнено успешно  IN\_PROGRESS – в процессе выполнения |
| activityTypeId | Идентификатор типа шага | start – стартовое событие  end – конечное событие  exclusiveGateway – эксклюзивный шлюз  web – пользовательская задача, отрабатываемая Модулем веб-интерфейса  script – скипт-задача, отрабатываемая Модулем исполнения скриптов  SEM\_LoаdScript – сервисная задача, отрабатываемая Модулем исполнения скриптов  MMM\_Model\_Import – сервисная задача, отрабатываемая Модулем управления моделями  MMM\_Model\_Validate – сервисная задача, отрабатываемая Модулем управления моделями |
| adapterTaskId | Идентификатор | Уникальное целое число |
| processPk | Идентификатор процесса | Уникальное целое число |

## Приложение №4

**Замечания ко всем API**

В описании API для каждого запроса указывается:

* *request* – запрос, посылаемый для выполнения соответствующей операции;
* *response* - ответ, получаемый в результате выполнения запроса.

*sessionId* - идентификатор, который ядро выдаёт адаптерам при регистрации. Необходим для исключения ошибок в следующих случаях:

1. Связь ядро - адаптер становится недоступной, ядро шлёт сообщение адаптеру, видит, что адаптер недоступен и исключает его из списка доступных адаптеров, после чего связь восстанавливается. В итоге адаптер будет “думать”, что подключён, однако ядро будет считать его недоступным.
2. Адаптер подключён к ядру, адаптер выключили без выполнения процедуры отмены регистрации, на том же endpoint запустили другой адаптер, который начал слать запросы без предварительной регистрации.
3. Появляются 2 адаптера с одинаковыми endpoint.

Ответ всегда может содержать поля:

* *response.status.type* - одно из значений: “SUCCESS” или “ERROR”;
* *response.status.code* - код ошибки, который может быть использован, например, для локализации;
* *response.status.params* - массив дополнительных параметров к ошибке;
* *response.status.message* - необязательное поле, сообщение, которое описывает ошибку в человекочитаемом виде.

*“instance”: {<>}* задаётся в виде json объекта, где названия полей соответствуют названиям полей в модели соответствующей версии

*entity.version* - версия модели. Необходимо иметь в виду, что:

* Версии одних и тех же сущностей могут быть разнесены по разным адаптерам;
* Методы приведения экземпляров сущностей к определённым версиям относятся к зоне ответственности адаптеров.

*request.paging* – запрашиваемая информация по постраничному выводу, где *scrollTo* - информация по тому, на какую страницу переходить, *pageSize* - максимальное количество записей в ответе

*response.data.rows* - поле для списка сущностей ответа при использовании постраничного вывода

*response.data.paging* – сводная информация по постраничному выводу, где totalRows - количество найденных записей, totalPages - количество доступных страниц, pageSize - размер страницы, pageNumber - текущий номер страницы, начинающийся с 1.

Каждый метод может вернуть ошибку с кодом ERROR\_SYSTEM.

**API ядра**

**registerAdapter**

Регистрирует адаптер в ядре.

request: {

“endpoint”: “<>”,

// Список активностей

“activityIds”: [

“<activityId>”,

...

],

// С какими сущностями умеет работать адаптер

“entities”: [{

“name”: “<entityName>”,

“versions”: [<>, ...]

}]

},

response: {

“data”: {

“sessionId”: “<>”

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**heartbeat**

Уведомление ядра о работоспособности адаптера.

Данный функционал может быть использован адаптерами для исключения ситуации, когда ядро посчитает адаптер отключённым, но сам адаптер об этом знать не будет и периодическая отсылка сообщений ядру не предусмотрена.

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“endpoint”: “<>”,

“sessionId”: “<>”

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**unregisterAdapter**

Сообщает ядру, что адаптер выключается.

request: {

“endpoint”: “<>”,

“sessionId”: “<>”

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**notifyTaskDone**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“adapterTaskPk”: {

“id”: <>,

“activityId”: “<>”

},

“status”: {

“type”: “SUCCESS” | “WARNING” | “ERROR”,

“code”: “<>”,

“message”: “<>”

},

“entities”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstancePk”: {“id”: <>}

}, ...]

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getEntityLocations**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“endpoint”: “<>”,

“sessionId”: “<>”,

“entities”: [{

“name”: “<>”,

“versions”: [<>, ...]

}, ...]

},

response: {

“data”: {

“entities”: [{

“name”: “<>”,

“versions”: [<>, ...],

“adapterEndpoints”: [“<>”, ...]

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**startEvent**

Запуск события

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“eventPk”: {“id”: <>}

},

response: {

“data”: {

“newProcessPk”: {“id”: <>},

“newProcessStepPk”: {“id”: <>}

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getProcesses**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“filters”: [{

“processPk”: {“id”: <>},

“createdBy”: “<>”,

“finished”: <bool>

}, …],

“paging”: {

scrollTo: {

“pageNumber”: <>,

“pk”: {“id”: <>}

},

“pageSize”: <>

}

},

response: {

“data”: {

“rows”: [{

“processPk”: {“id”: <>},

“poolPk”: {“id”: <>},

“name”: “<>”,

“startTime”: <>,

“endTime”: <>,

“createdBy”: “<>”,

“processStepPks”: [{

“pk”: {“id”: <>}

}, ...]

}, ...],

“paging”: {

“totalRows”: <>,

“totalPages”: <>,

“pageSize”: <>,

“pageNumber”: <>

}

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getProcessSteps**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“filters”: [{

“pk”: {<>}

}, ...],

“paging”: {

scrollTo: {

“pageNumber”: <>,

“processStepPk”: {<>}

},

“pageSize”: <>

}

},

response: {

“data”: {

“rows”: [{

“processPk”: {“id”: <>},

“flowNodePk”: {“id”: <>},

“adapterEndpoint”: “<>”,

“adapterTasktPk”: {“id”: <>},

“startTime”: <>,

“endTime”: <>,

“adapterTaskStatus”: “SUCCESS” | “WARNING” | “ERROR”

“createdBy”: “<>”,

“updatedBy”: “<>”

}, ...],

“paging”: {

“totalRows”: <>,

“totalPages”: <>,

“pageSize”: <>,

“pageNumber”: <>

}

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getPools**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“filters”: [{

“poolPk”: {“id”: <>}

}, ...]

},

response: {

“data”: {

“rows”: [{

“poolPk”: {“id”: <>},

“name”: “<>”

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

removeProcesses

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“processPks”: [{“id”: <>}, ...]

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getEvents**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“paging”: {

scrollTo: {

“pageNumber”: <>,

“pk”: {“id”: <>}

},

“pageSize”: <>

}

},

response: {

“data”: {

“rows”: [{

“pk”: {<>},

“name”: “<>”

}, ...],

“paging”: {

“totalRows”: <>,

“totalPages”: <>,

“pageSize”: <>,

“pageNumber”: <>

}

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getFlowNodes**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“paging”: {

scrollTo: {

“pageNumber”: <>,

“pk”: {“id”: <>}

},

“pageSize”: <>

},

“filters”: [{

}, ...]

},

response: {

“data”: {

“rows”: [{

“pk”: {“id”: <>},

“type”: “ACTIVITY”, “EVENT”, “GATEWAY”,

“activityId”: “”,

“attributes”: {key-value-pairs}

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getEntityDescriptions**

Коды ошибок:

● ERR\_SESSION\_NOT\_EXISTS

request: {

“authContext”: {

“login”: “<>”,

“isSuperUser”: <bool>,

“permissions”: [“<>”, ...]

},

“endpoint”: “<>”,

“sessionId”: “<>”,

“entities”: [{

“name”: “<>”,

“version”: <>

}, ...]

},

response: {

“data”: {

“rows”: [{

“pk”: {

“name”: “<>”,

“version”: <>

},

“fields”: [{

“pk”: {“id”: <>},

“name”: “<>”,

“minInstances”: <>,

“maxInstances”: <>,

“type”: “<>”,

“linkedEntityPk”: {“id”: <>},

“defaultValue”: “<>”,

“constraint”: “<>”,

“constraintHint”: “<>”,

“format”: “<>”,

“order”: <>,

“isForEntityTitle”: <bool>

}, ...]

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**API адаптера**

**createNewTask**

Отправляет запрос в адаптер.

adapter.requestNumber - id запроса в ядре

adapter.activityId необходим, т.к. адаптер может выполнять сразу несколько активностей

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“adapterTaskPk”: {

“id”: <>,

“activityId”: “<>”,

},

“process”: {

“processStepPk: {“id”: <>},

“createdBy”: “<>”

},

“inputEntityInstances”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstancePk”: {“id”: <>}

}, ...],

“outputEntities”: [{

“name”: “<>”,

“version”: <>

}, ...]

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**createEntityInstances**

Создаёт сущности entities.instance, заполняет все поля, которые автоматически заполняются адаптером, например, уникальный сквозной идентификатор сущности, и возвращает сохранённые сущности.

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“transactionId”: “<>”,

“entities”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstance”: {<>}

}, ...]

},

response: {

“data”: {

“rows”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstance”: {<>}

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**readEntityInstances**

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“transactionId”: “<>”,

“entityInstancePks”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstancePk”: {“id”: <>}

}, ...]

},

response: {

“data”: {

“rows”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstance”: {<>}

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**updateEntityInstances**

Обновляет сущности. Если хоть одну из сущностей обновить не удалось, не обновляет ни одной.

В ответе либо заполняется поле response.data.rows.instances, если все сущности обновлены успешно, либо возвращается заполненное поле response.data.rows.validationErrors если возникли ошибки при валидации сущности.

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“lockId’: “<>”,

“transactionId”: “<>”,

“entities”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstance”: {<>}

}, ...]

},

response: {

“data”: {

“rows”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstance”: {<>},

“validationErrors”: [{

“entityInstancePk”: {“id”: <>},

“code”: “<>”,

“params”: [“”, ...]

}, ...]

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**deleteEntityInstances**

Удаляет экземпляры сущностей. Если хотя бы одна из сущностей не может быть удалена не удаляет ни одной.

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“lockId’: “<>”,

“transactionId”: “<>”,

“entityInstancePks”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstancePk”: {“id”: <>}

}, ...]

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**openTransaction**

Открывает транзакцию с уровнем изоляции READ\_COMMITED. Если транзакция не будет подтверждена или отклонена в течении rollbackTimeoutSecs, то она будет автоматически отменена. Значение по-умолчанию для rollbackTimeoutSecs задаётся в настройках адаптера.

Коды ошибок:

● ERR\_TRANSACTION\_NOT\_SUPPORTED

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“rollbackTimeoutSecs”: <>

},

response: {

“data”: {

“transactionId”: “<>”

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**prolongateTransaction**

Продлевает действие транзакции на rollbackTimeoutSecs с момента получения prolongate-transaction, забывая предыдущее значение.

Коды ошибок:

● ERR\_TRANSACTION\_NOT\_SUPPORTED

● ERR\_TRANSACTION\_NOT\_OPENED

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“transactionId”: “<>”,

“rollbackTimeoutSecs”: <>

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**commitTransaction**

Подтверждает транзакцию.

Коды ошибок:

● ERR\_TRANSACTION\_NOT\_SUPPORTED

● ERR\_TRANSACTION\_NOT\_OPENED

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“transactionId”: “<>”

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**rollbackTransaction**

Отменяет транзакцию.

Коды ошибок:

● ERR\_TRANSACTION\_NOT\_SUPPORTED

● ERR\_TRANSACTION\_NOT\_OPENED

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“transactionId”: “<>”

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**lockEntityInstances**

В запросе может быть указан либо lockId либо entities.

Если заполнен entities, то будет произведён запрет на изменение и удаление экземпляров сущностей. Либо все экземпляры сущностей удаётся заблокировать, либо блокировка не происходит совсем. Результатом блокировки является lockId, по которому можно будет разлочить все сущности. Автоматическая разблокировка сущностей будет произведена через unlockTimeoutSecs секунд.

Значение поля unlockTimeoutSecs не обязательно для заполнения и его значение по умолчанию задаётся настройками адаптера.

Коды ошибок:

● ERR\_LOCK\_NOT\_SUPPORTED

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“unlockTimeoutSecs”: <>,

“entitiyInstancePks”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstancePk”: {“id”: <>}

}, ...]

},

response: {

“data”: {

“lockId”: “<>”

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**prolongateLock**

Пролонгирование существующей блокировки на unlockTimeoutSecs секунд.

Коды ошибок:

● ERR\_LOCK\_NOT\_SUPPORTED

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“unlockTimeoutSecs”: <>,

“lockId”: “<>”

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**unlockEntityInstances**

Разблокировка блока lockId. Если хоть одну из сущностей из блокировки lockId не удастся разблокировать, то и остальные разблокированы не будут.

Коды ошибок:

● ERR\_LOCK\_NOT\_SUPPORTED

● ERR\_LOCK\_NOT\_EXIST

request: {

“userLogin”: “<>”,

“sessionId”: “<>”,

“lockId”: “<>”

},

response: {

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}

**getLockedEntityInstances**

Коды ошибок:

● ERR\_LOCK\_NOT\_SUPPORTED

request: {

“userLogin”: “<>”,

“sessionId”: “<>”

},

response: {

“data”: {

“rows”: [{

“lockId”: “<>”,

“entityInstancePks”: [{

“entityPk”: {

“name”: “<>”,

“version”: <>

},

“entityInstancePk”: {“id”: <>}

}, ...]

}, ...]

},

“status”: {

“type”: “SUCCESS” | “ERROR”,

“code”: “<>”,

“params”: [“”, ...],

“message”: “<>”

}

}