# СИСТЕМА ОРКЕСТРАЦИИ

## Описание продукта

Версия 9.2.2

nexign

АО «Нэксайн»

Россия, 199155, Санкт-Петербург, ул. Уральская, д.4 лит.Б, помещение 22Н

Тел.: + 7 (812) 326-12-99; факс: + 7 (812) 326-12-98.

office@nexign.com; www.nexign.com

# Содержание

# 1. Назначение

Продукт «Система оркестрации (Nexign Orchestration Engine, **CRAB**)» версии 9.2.2 (далее – **CRAB**) является системой исполнения интеграционных сценариев и позволяет автоматизировать процессы в распределенной информационной среде, когда взаимодействующие программные продукты установлены на разных серверах, филиалах и центрах обработки данных. Сценарии могут быть связаны как с решением бизнес-задач (например, продажа и подключение абоненту услуги), так и быть исключительно техническими (например, передача данных на оборудование).

# 2. Ключевые преимущества

## 2.1. Асинхронное исполнение интеграционных сценариев

Асинхронное исполнение позволяет автоматизировать процессы, продолжительность которых составляет от миллисекунд до нескольких месяцев.

## 2.2. Поддержка широкого спектра протоколов взаимодействия с системами, участвующими в автоматизируемом процессе

Библиотека **ApacheCamel**, используемая для реализации шагов интеграционных сценариев, позволяет автоматизировать процессы, в которых участвуют системы, предоставляющие свой API по различным протоколам (HTTP, AMQP и т.д.).

## 2.3. Исполнение сценариев с учетом зависимостей

Возможность указывать при регистрации заявок зависимости по результату между заявками позволяет автоматизировать запуск процессов, исполнение которых необходимо только при определенных условиях.

## 2.4. Последовательный доступ к заданному ресурсу

Специальный последовательный тип обработки заявок, указываемый при регистрации заявок совместно с абстрактным именем ресурса, позволяет избегать дополнительных затрат по обеспечению монопольного доступа к заданному ресурсу. В качестве имени ресурса (в терминах **CRAB** – *имя очереди*) можно использовать:

- доменное имя сервера с внешней системой;
- экземпляр сущности доменной модели (конкретного клиента, абонента и т.п.).

## 2.5. Максимизация успешного исполнения сценариев

Высокий уровень успешного исполнения сценариев обеспечивается:

- гибкой настройкой классификации ошибок взаимодействия с внешними системами;
- применением политик автоматического повтора исполнения.

Автоматическое замедление/приостановка потоков заявок при деградации/недоступности внешних систем позволяет работать без перегрузки на предельно допустимом уровне их производительности. Как следствие – минимизируется количество исполненных с ошибками сценариев в тех случаях, когда:

- время деградации/недоступности превышает период работы политик повтора исполнения;
- или для таких ошибок не настроены правила классификации.

## 2.6. Эффективная утилизация ограниченных ресурсов

Используемые алгоритмы распределения заявок и универсальные исполнители сценариев обеспечивают высокую утилизацию ресурсов:

- распределение в соответствии с приоритетами заявок позволяет максимизировать скорость обработки критичных процессов;
- распределение в соответствии с нормами потоков заявок позволяет использовать все свободные ресурсы для обработки пиковой нагрузки по отдельным потокам без влияния на остальные;
- управление лимитами потоков заявок «сглаживают» пиковые нагрузки на внешние системы.

## 2.7. Высокая производительность и отказоустойчивость

Горизонтальное масштабирование системы позволяет справиться с возрастающей нагрузкой при росте абонентской базы и появлении новых автоматизированных процессов.

Размещение резервных узлов продукта вплоть до отдельного центра обработки данных (ЦОД) обеспечивает устойчивость к отказу как отдельных серверов, так и ЦОД в целом.

# 3. Функциональные возможности

**CRAB** предоставляет возможности:

- исполнения интеграционных сценариев;
- интеграции с внешними системами;
- одновременного обслуживания разных процессов;
- решения задач по эксплуатации продукта, в том числе с использованием графического пользовательского интерфейса (UI).

## 3.1. Исполнение интеграционных сценариев

*Сценарии* – средство автоматизации процессов какой-либо предметной области. Процесс представляет собой набор определённых шагов и порядок их прохождения. Сценарий, как отражение процесса, также состоит из набора шагов и операций, выполняемых на каждом шаге. При исполнении сценария продукт выполняет логику, заданную в текущем шаге, и на основании его результата определяет следующий для выполнения шаг.

Сценарии **CRAB** представляют собой типовую машину состояний. Состояния соответствуют шагу, а действия, связанные с состояниями — операциям (Рис. 1).

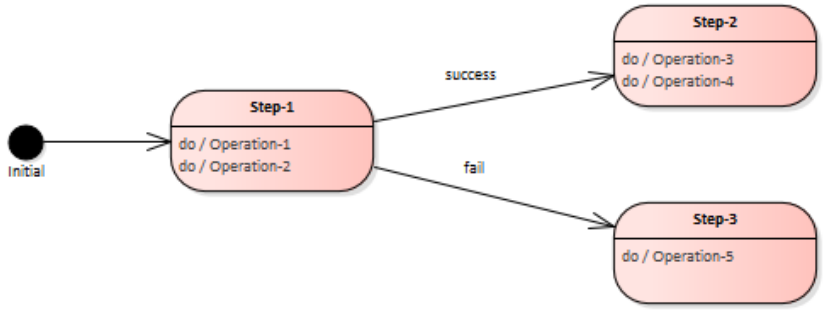


Рис. 1. Логическая схема исполнения сценария

### Управление порядком исполнения шагов сценария

Исполнение шага сценария заключается в исполнении заданных в нём операций. Операции одного шага выполняются последовательно, друг за другом.

Порядок исполнения шагов сценария, как правило, является нелинейным. Спецификация сценария задаёт возможные пути исполнения в виде условий выбора следующего шага сценария в зависимости от результата завершения текущего. В момент исполнения продукт фиксирует результаты шагов и выбирает следующий шаг в соответствии со спецификацией. Логика интеграционного сценария, определяемая через спецификацию, позволяет реализовать такие типовые управляющие конструкции как ветвления и циклы.

### Взаимодействие сценария с внешними системами

Практически каждый интеграционный сценарий в процессе исполнения взаимодействует с разными внешними системами для выполнения задач автоматизированного процесса.

Сценарии поддерживают следующие типовые шаблоны (паттерны) взаимодействия:

**модель «Запрос-ответ» (Request-Response):**

- для синхронного взаимодействия, например, по HTTP-протоколу;

- для асинхронного взаимодействия, когда обработка запроса внешней системой занимает продолжительное время;

**модель «Опрос» (Polling):**

- для случаев, когда внешняя система самостоятельно не возвращает результат операции;
- для случаев, когда для продолжения исполнения сценария необходимо дождаться определённого события во внешней системе.

При асинхронном взаимодействии и при взаимодействии по модели «Опрос» исполнение сценария приостанавливается после отправки запроса на внешнюю систему. Возобновление исполнения происходит при получении ответа внешней системы, по истечении заданного периода ожидания ответа или при наступлении времени следующей попытки опроса.

### Приостановка исполнения сценария

**CRAB** предоставляет возможность выполнять приостановку исполнения сценария, не связанную с ожиданием выполнения какой либо операции во внешней системе. В этом случае приостановка исполнения возможна на длительный период с использованием специальных шагов приостановки. Возобновление исполнения сценария происходит автоматически в указанное сценарием время при переходе в состояние приостановки.

### Реакция на внешние события

Выбор следующего шага для исполнения может происходить не только на основании результатов исполнения предыдущего шага, но и на основании зарегистрированного события, являющегося «внешним» по отношению к сценарию. Внешние события регистрируются через API **CRAB**.

## 3.2. Интеграция систем с продуктом CRAB

Интеграция систем с продуктом **CRAB** выполняется с целью автоматического исполнения определенных процессов и заключается во взаимодействии систем с продуктом через его API.

### 3.2.1. Исполнение сценариев по заявкам систем

Исполнение интеграционных сценариев продуктом выполняется по запросам систем-клиентов. Каждый такой запрос на исполнение сценария в **CRAB** рассматривается как отдельная команда и называется *заявкой*.

#### Регистрация заявок по синхронному и асинхронному каналам

Продукт предоставляет возможность регистрировать заявки на исполнение сценария:

- по синхронному каналу с использованием REST API;
- по асинхронному каналу с использованием AMQP API.

Продукт также предоставляет API для регистрации нескольких заявок в одном запросе как для синхронного канала, так и для асинхронного.

При регистрации заявки система может передать набор параметров, необходимых сценарию для его исполнения.

#### Динамическое определение сценария при создании заявки

Полная спецификация сценария, содержащая описание набора шагов сценария и правила

перехода между ними, задаётся в момент регистрации заявки. С этого момента спецификация хранится без изменений на протяжении всего жизненного цикла заявки. Такой подход:

- позволяет определять сценарий непосредственно в момент регистрации заявки;
- гарантирует неизменность сценария на протяжении всего жизненного цикла заявки.

Наиболее ярко данный подход проявляет себя при интеграции с системами, способными динамически формировать спецификацию с учётом условий исполняемого процесса.

В качестве альтернативного варианта **CRAB** предоставляет возможность регистрации заявок с использованием одного из своих компонентов – каталога *workflow*. В этом случае спецификация сценария выбирается из заранее определённого списка спецификаций на основании заданных параметров запроса.

### Асинхронное исполнение сценариев

Исполнение сценария происходит асинхронно. Это означает, что на запрос системы на исполнение сценария продукт отвечает подтверждением успешной регистрации заявки. Результат исполнения сценария будет отправлен после полного завершения работы сценария.

### Ограничение периода до старта исполнения

Иногда начало исполнения сценария задерживается — при вбросах больших пачек новых заявок, при приостановке потока в связи с недоступностью систем и т.п. Для некоторых процессов исполнение сценария после определённого момента времени может быть недопустимым.

**CRAB** предоставляет возможность указать при регистрации заявки максимальный период времени, отведённый на начало исполнения сценария. Если такая заявка не перешла на этап исполнения сценария до истечения указанного периода по какой-либо причине, **CRAB** завершает её обработку как неуспешную.

### Нотификация о результатах исполнения

Обработка заявки на исполнение сценария может завершиться с разным результатом:

- сценарий полностью исполнен;
- исполнение сценария было прервано в результате критичной ошибки;
- сценарий не может быть исполнен по разным причинам: был отменен, истек период начала исполнения и т.п.

При любом исходе продукт отправляет сообщение с результатами обработки заявки. Необходимость отправки результатов и адресат задаются параметрами при регистрации заявки.

## 3.2.2. Определение порядка исполнения сценариев

Для одних процессов может быть критичен порядок исполнения, а для других важно избежать конкурентного изменения какого-либо ресурса во внешней системе.

Продукт предоставляет возможность при регистрации заявки определять условия начала исполнения сценария: задавать тип обработки и/или указывать зависимости между заявками.

### Тип обработки

Существует два типа обработки заявок – *параллельная* и *последовательная*. Тип обработки указывает продукту на взаимный порядок исполнения сценариев.

Заявки с параллельным типом обработки переходят на стадию исполнения сценария сразу

после регистрации.

Заявки с последовательным типом исполняются строго последовательно, если они находятся в одной очереди. Очередь указывается при регистрации заявки и представляет собой абстрактный идентификатор, практический смысл которого определяется системой, регистрирующей заявку. Заявки, имеющие одинаковое значение этого идентификатора, относятся к одной очереди.

**Зависимая обработка заявок**

При регистрации заявки система может указать её зависимость от результатов исполнения других сценариев. Указанная зависимость фактически является условием начала исполнения текущего сценария.

Использование механизма зависимостей между заявками позволяет:

- обрабатывать процессы, шаги которых зависимы по результатам работы, но реализуются разными сценариями; например, если один сценарий создаёт некоторую сущность, а другой её изменяет, то зависимые заявки позволяют начать изменение сущности только после её успешного создания;
- соблюдать строгий порядок обработки ряда событий; например, если при изменении некоторой сущности регистрируется заявка по обработке данного события, то зависимые заявки позволяют гарантировать последовательность обработки событий без пропусков (даже в случае возникновения ошибок);
- динамически расширять бизнес-сценарии; например, если в обработку была принята предусловная заявка в рамках некоторого бизнес-сценария, то условные заявки позволяют динамически расширить бизнес-сценарий дополнительными шагами в случае успешной или даже неуспешной обработки предусловной заявки.

## 3.3. Эффективное исполнение сценариев в условиях ограниченных ресурсов

При эксплуатации системы периодически возникают ситуации, когда для немедленного исполнения всех зарегистрированных заявок текущих ресурсов (как внутренних, так и внешних) недостаточно. Причины могут быть самые разные:

- остановка части серверов для технического обслуживания;
- вброс большой пачки заявок по одному или нескольким процессам;
- деградация внешних систем и т.д.

Продукт предоставляет возможности наиболее эффективного использования ресурсов исполнения в подобных ситуациях.

### 3.3.1. Приоритет заявок

При регистрации заявки система может указать приоритет её исполнения в соответствии с критичностью процесса. Для высоко критичных процессов необходимо установить высокий приоритет, для фоновых задач, не требующих срочности исполнения, – низкий.

Для исполнения сценариев по заявкам с разным приоритетом продукт выделяет разные доли ресурсов исполнения. Чем выше приоритет, тем большая доля ресурсов отводится для исполнения.

В то же время, если по группе заявок какого-либо приоритета есть избыточные ресурсы, они будут использованы для исполнения сценариев по заявкам другого приоритета.

### 3.3.2. Распределение ресурсов исполнения между потоками заявок

Весь поступающий объем заявок разделяется на разные потоки в соответствии с именем сценария. Для каждого потока на основании долгосрочной статистики рассчитывается необходимая доля ресурсов исполнения.

При вбросе большой пачки заявок по одному или нескольким потокам продукт обеспечивает сохранение объема ресурсов исполнения для остальных потоков в пределах их долей. Этот механизм не допускает снижения производительности обработки одних потоков при резком увеличении количества заявок по другим.

Продукт предоставляет дополнительные возможности по разделению заявок на потоки в соответствии с дополнительными атрибутами. Это полезно, когда нужно, например, изолировать обработку заявок по одному и тому же сценарию для разных регионов.

### 3.3.3. Управление лимитами для потоков заявок

В момент «всплесков» количества заявок в потоках внешние системы, с которыми взаимодействуют сценарии, испытывают повышенную нагрузку. В ряде случаев это приводит к их деградации и даже полному отказу.

Продукт **CRAB** имеет встроенный механизм защиты внешних систем от чрезмерной нагрузки со стороны сценариев. Продукт предоставляет возможность автоматического управления количеством одновременно исполняющихся сценариев по заявкам одного потока – лимитирование потоков. Для каждого из потоков на основании статистики исполнения рассчитывается его лимит. Заявка по потоку переходит на этап исполнения сценария только в том случае, если текущее количество исполняющихся сценариев по этому потоку не достигло установленного лимита.

Для автоматического управления лимитом потоков продукт **CRAB** анализирует характеристики потоков – длительность исполнения сценариев и количество ошибок. При деградации или недоступности внешней системы длительность исполнения сценариев увеличивается, а количество ошибок возрастает. В этом случае продукт снижает лимит по потоку. После нормализации ситуации продукт постепенно возвращает значение лимита к необходимому уровню.

## 3.4. Эксплуатационные возможности

В процессе эксплуатации продукта возникает необходимость контроля и анализа:

- процесса обработки отдельных заявок;
- работоспособности его функций в целом.

### 3.4.1. Контроль обработки заявок

Для контроля обработки отдельных заявок в продукте предусмотрены:

- графический пользовательский интерфейс с основной информацией по заявке, порядком исполнения шагов, их результатами и результатом исполнения сценария;
- система журналирования с информацией о деталях процесса обработки;
- возможность контролировать состояние ядра **CRAB** на одной площадке;
- управление заявками на выполнение бизнес-сценариев.

### 3.4.2. Контроль работоспособности функций

Для контроля и анализа состояния и работы функций продукта предусмотрены:

- графический интерфейс с информацией о состоянии узлов продукта;
- система мониторинга;
- система журналирования.

# 4. Основные модели данных

## Логическая модель CRAB_AKKA

Схема модели данных **CRAB_AKKA** представлена на Рис. 2.

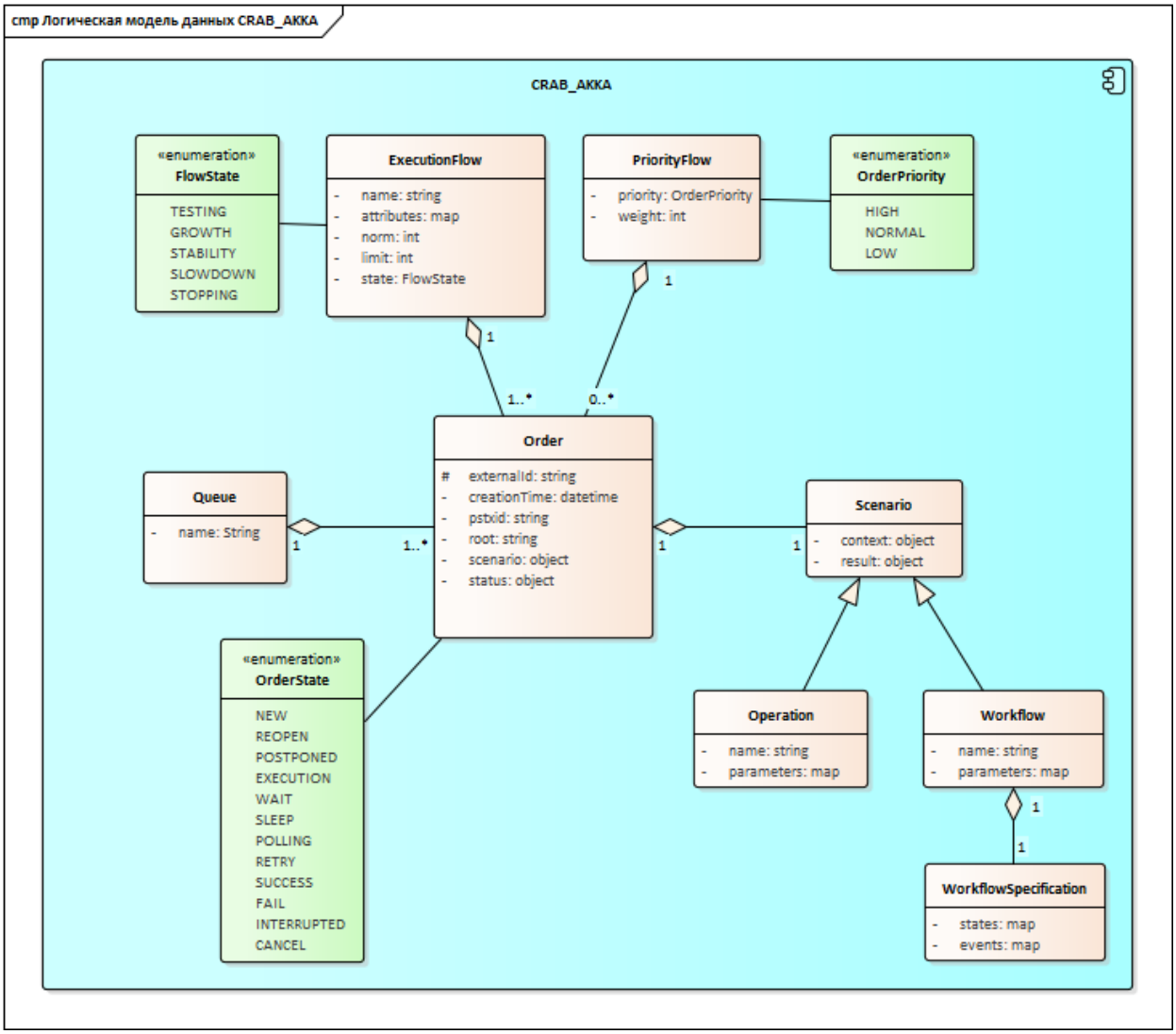


Рис. 2. Схема модели данных CRAB_AKKA

Описание классов логической модели **CRAB_AKKA** приведено в Табл. 1.

Табл. 1. Классы логической модели **CRAB_AKKA**

| Класс | Описание |
|---|---|
| `Order` | Заявка на исполнение сценария. Основная сущность, отражающая запрос системы-клиента на исполнение сценария |
| `PriorityFlow` | Поток одноприоритетных заявок – множество заявок с одним уровнем срочности исполнения сценария |

| Класс | Описание |
|---|---|
| `ExecutionFlow` | Поток исполнения заявок – множество заявок на выполнение однотипной работы с помощью одного состава исполнителей. Под однотипностью работы понимается множество выполняемых операций при исполнении сценария. Под одним составом исполнителей понимается набор внешних систем, с которыми взаимодействует сценарий. Заявки на исполнение разных сценариев автоматически относятся к разным потокам. Заявки по исполнению одного и того же сценария могут быть отнесены к разным потокам, если часть операций исполняется только для одного из подмножества заявок (так называемые опциональные шаги) |
| `Queue` | Очередь заявок, по которым исполнение сценария требует монопольного доступа к внешнему ресурсу. Под внешним ресурсом может пониматься клиент, абонент, оборудование и т.п. |
| `Scenario` | Сценарий, реализующий какой-либо процесс |
| `Operation` | Сценарий, в котором шаги, их порядок и логика описываются с помощью **Apache Camel Groovy DSL** |
| `Workflow` | Сценарий, в котором шаги и их порядок описываются workflow-спецификацией, а логика – средствами **Apache Camel Groovy DSL** |
| `WorkflowSpecification` | Спецификация workflow – описание шагов сценария и их порядка |

# 5. Функциональная архитектура

Общая компонентная схема **CRAB** представлена на Рис. 3.

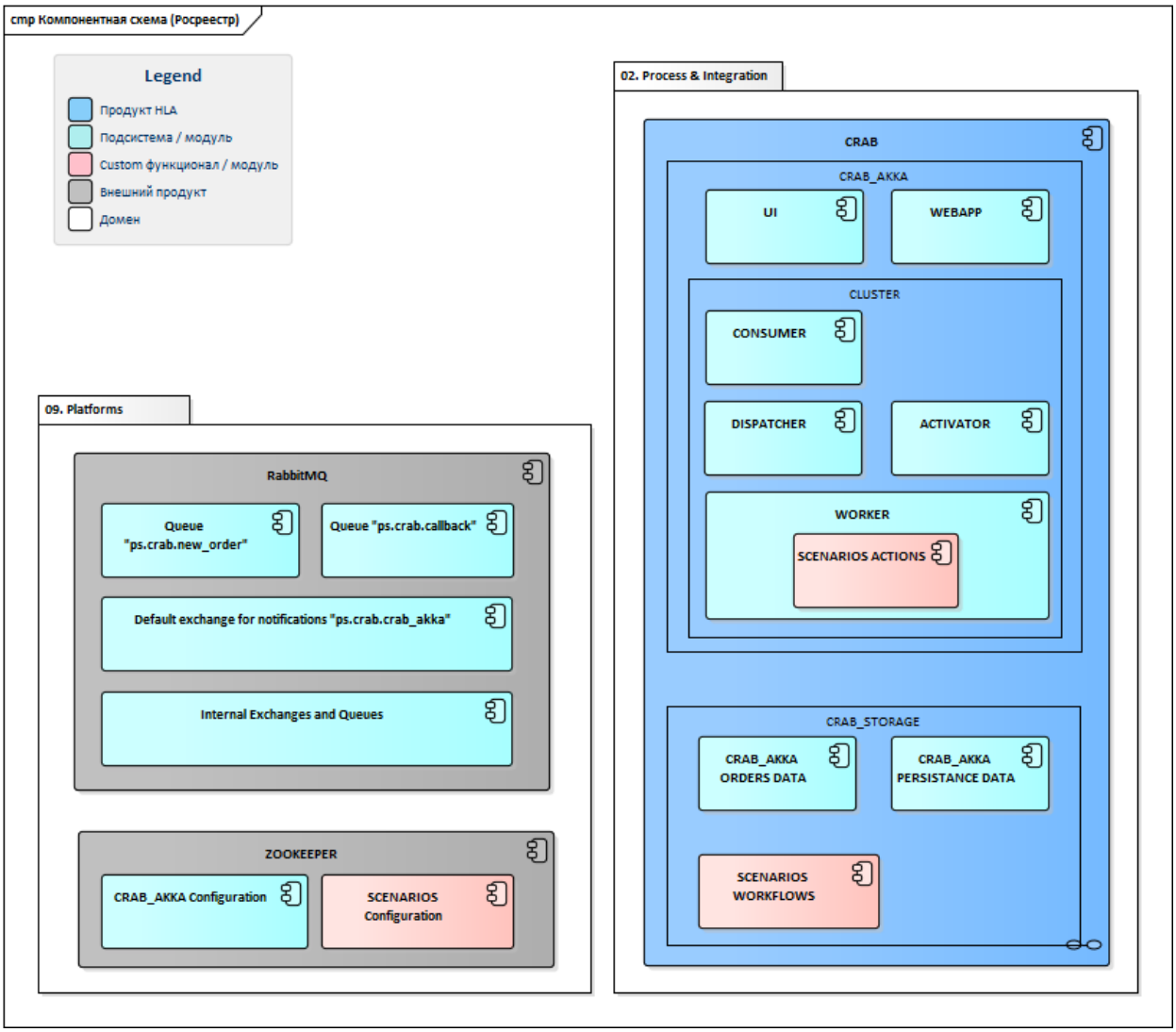


Рис. 3. Общая компонентная схема CRAB

Описание компонентов **CRAB** и их модулей приведено в Табл. 2.

Табл. 2. Компоненты CRAB

| **Компоненты** | **Модули** | **Описание** |
|---|---|---|

| | | | |
|---|---|---|---|
| **CRAB_AKKA** | **UI** | | Компонент предоставляет графический пользовательский интерфейс:<br>• для работы с заявками;<br>• с информацией о состоянии узлов продукта |
| | **WEBAPP** | | Компонент предоставляет REST API внешним потребителям:<br>• для работы с заявками;<br>• получения информации о состоянии узлов продукта |
| | **CLUSTER** | **CONSUMER** | Компонент предоставляет AMQP API внешним потребителям:<br>• по работе с заявками;<br>• по работе с ответными (callback) сообщениями |
| | | **DISPATCHER** | Компонент:<br>• управляет потоками заявок;<br>• распределяет заявки по исполнителям в соответствии с очерёдностью, приоритетом и другими факторами |
| | | **ACTIVATOR** | Компонент активирует обработку приостановленных заявок после истечения времени приостановки |
| | | **WORKER** | Компонент:<br>• исполняет сценарии;<br>• отправляет нотификации о завершении их исполнения |

# 6. Настройка и функциональное расширение

## 6.1. Автоматизация клиентских процессов

Для решения своей основной задачи – исполнения автоматизированных процессов продукт использует интеграционные сценарии. Перед непосредственным исполнением разработанные сценарии должны быть установлены в развернутый на определенной площадке продукт **CRAB**.

### 6.1.1. Разработка интеграционных сценариев

Разработка интеграционного сценария состоит:

- из описания спецификации;
- разработки отдельных операций, реализующих логику работы шагов процесса.

Входными артефактами для разработчика сценария являются:

- диаграммы последовательности с описанием автоматизируемого процесса;
- API и модели данных систем, участвующих в автоматизируемом процессе.

Результатом разработки сценария является подготовленный набор:

- файлов спецификаций;
- исполняемых файлов с операциями.

#### Спецификация сценария

Спецификация сценария представляет собой JSON структуру, которая определяет набор шагов сценария и порядок перехода между ними. В зависимости от способа интеграции спецификация сценария может передаваться клиентом при создании новой заявки или выбираться из каталога *workflow* на основании параметров запроса клиента.

Ниже приведён пример спецификации простого сценария:

```
"workflow": {
  "name": "dummyWorkflow",
  "initialState": "INITIAL_STATE",
  "states": {
    "INITIAL_STATE": {"actions": [{"name": "dummy"}], "terminal":
false},
    "TERMINAL_STATE": {"actions": [], "terminal": true}
  },
  "events": {
    "SUCCESS": [{"from": "INITIAL_STATE", "to": "TERMINAL_STATE"}]
  },
  "parameters": {
    "subsId": "1"
  }
}
```

### Операции шагов сценария

Шаги сценария реализуют логику исполнения шагов процесса. Каждый шаг сценария может содержать одну или несколько операций.

Как правило, каждая операция имеет одно из следующих назначений:

- подготовка данных;
- взаимодействие с внешней системой;
- управление порядком исполнения;
- формирование результата сценария.

Атомарность назначения позволяет повторно использовать одну и ту же операцию как в разных шагах одного сценария, так и в разных сценариях.

Операция представляет собой объект маршрута библиотеки **ApacheCamel**, реализованный с использованием **Apache Camel Java DSL**. Интерфейс взаимодействия сценария с продуктом **CRAB** представляет собой набор предопределенных **ApacheCamel**-компонентов.

Итоговые скомпилированные файлы операций компонуются в jar-файлы.

### Переиспользование функциональности

Переиспользование реализованной функциональности может быть выполнено на разных уровнях:

- **компонент Apache Camel**. В рамках одного решения большинство систем поддерживают единые правила интеграции. В этом случае разработчик сценариев может реализовать компонент, поддерживающий данные правила и использовать его в разных операциях для взаимодействия с системами решения;
- **операция**. Одна и та же операция может быть использована в разных шагах одного сценария, а так же в шагах других сценариев. Например, операция проверки баланса;
- **сценарий**. Два и более процесса могут иметь общую логику или включать один и тот же «подпроцесс». Например, последовательность шагов по подключению/отключению услуги в рамках процессов регистрации нового абонента и финансовой блокировки/разблокировки. В этом случае такую общую последовательность шагов можно реализовать в виде отдельного сценария. Верхнеуровневые сценарии могут регистрировать новую заявку на исполнение «дочернего» сценария и ожидать его результата, аналогично кейсу асинхронного взаимодействия.

## 6.1.2. Развертывание интеграционных сценариев

Процедура развертывания интеграционных сценариев включает:

- установку файлов с операциями на узлы исполнителей продукта **CRAB**;
- установку файлов спецификаций в зависимости от схемы интеграции с системами решения:
  - на узлы с workflow-каталогом;
  - либо на узлы системы, регистрирующей новые заявки в **CRAB**;
- установку параметров сценариев в ZooKeeper (при их наличии);
- перезагрузку узлов исполнителей и узлов с workflow-каталогом для активации сценариев.

Для развертывания интеграционных сценариев, разрабатываемых в **Nexign**, используется встроенный автоинсталлятор.

# 6.2. Настройка продукта

Настройка продукта **CRAB** производится с целью:

- обеспечения работы продукта в определенном окружении;
- оптимизации работы функций продукта в соответствии со специфичным профилем нагрузки.

Настройка окружения включает в себя установку параметров работы:

- с хранилищами заявок и заказов;
- хранилищем состояния распределителя;
- брокером сообщений.

Настройка функций продукта включает в себя установку параметров:

- функции распределения заявок;
- функции исполнения заявок;
- функции активации заявок;
- обработки ошибок взаимодействия:
  - параметров классификации ошибок;
  - параметров повтора взаимодействий.

# 7. Интеграция

Глава включает описание:

- схемы взаимодействия **CRAB** с внешними системами;
- интеграционных интерфейсов;
- REST API **CRAB**.

## 7.1. Схема взаимодействия

Состав интеграций **CRAB** с внешними продуктами определяется исполняемыми сценариями. Примерная схема интеграции представлена на Рис. 4.

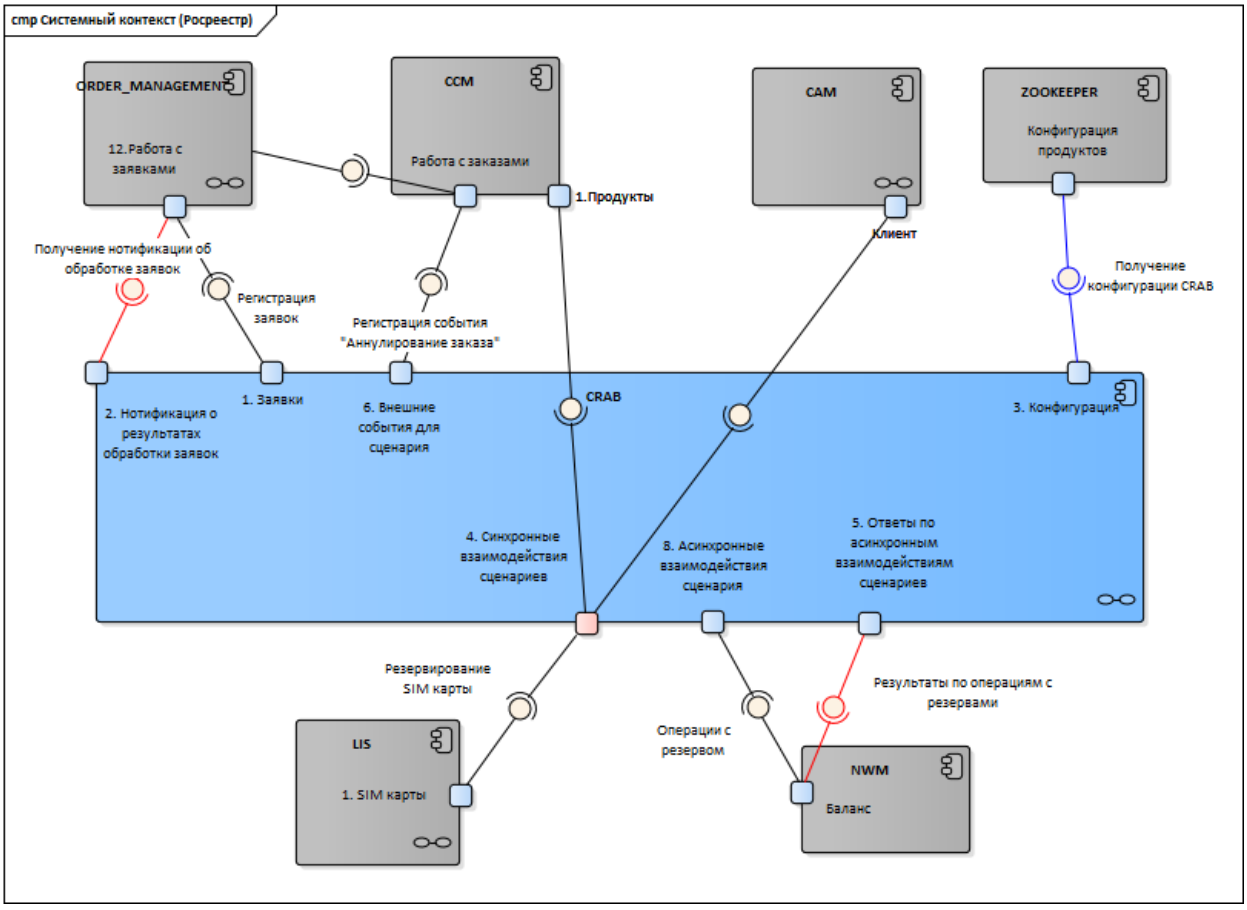


Рис. 4. Примерная схема интеграции CRAB с внешними продуктами

На диаграмме продукты **ORDER_MANAGEMENT**, **CCM**, **CAM**, **LIS** и **NWM** приведены только для примера интеграций.

## 7.2. Интеграционные интерфейсы

### 7.2.1. Описание интеграций (провайдер)

| № | Группа интерфейсов (порт) | Описание | Детали взаимодействия |
|---|---|---|---|
| 1 | Заявки | Операции с заявками | Формат взаимодействия: JSON over HTTP/AMQP. Режим работы: синхронный/асинхронный. Формат доставки: гарантия доставки обеспечивается повторными запросами со стороны клиента |
| 5 | Ответы по асинхронным взаимодействиям сценариев | Регистрация ответа по асинхронному взаимодействию сценария и внешней системы | Формат взаимодействия: JSON over AMQP. Режим работы: асинхронный. Формат доставки: гарантия доставки обеспечивается повторными асинхронными вызовами со стороны сценария |
| 6 | Внешние события для сценария | Регистрация внешних событий по сценарию | Формат взаимодействия: JSON over HTTP. Режим работы: синхронный (в отношении регистрации события). Формат доставки: гарантия доставки обеспечивается повторными запросами со стороны клиента |

## 7.2.2. Описание интеграций (потребитель)

| № | Группа интерфейсов (порт) | Провайдер | Описание | Детали взаимодействия |
|---|---|---|---|---|
| 2 | Нотификация о результатах обработки заявок | Внешние системы, ожидающие результаты обработки заявки | Нотификация о результате обработки заявки. Если при обработке исполнялся сценарий, то результаты обработки содержат результат исполнения сценария | Формат взаимодействия: JSON over HTTP/AMQP. Режим работы: синхронный/асинхронный |
| 3 | Конфигурация | ZooKeeper | Конфигурирование продукта | Формат взаимодействия: native. Режим работы: синхронный |
| 4 | Синхронные взаимодействия сценариев | Внешние системы, с которыми взаимодействует сценарий синхронно | Взаимодействие с внешними системами при исполнении сценария. Результат взаимодействия возвращается синхронно | Формат взаимодействия: JSON over HTTP/AMQP. Возможны другие форматы и протоколы, доступные в **ApacheCamel**. Режим работы: синхронный |
| 8 | Асинхронные взаимодействия сценариев | Внешние системы, с которыми взаимодействует сценарий асинхронно | Взаимодействие с внешними системами при исполнении сценария. Результат взаимодействия возвращается асинхронно | Формат взаимодействия: json over HTTP/AMQP. Возможны другие форматы и протоколы, доступные в **ApacheCamel** Режим работы: асинхронный |

### 7.2.3. Перечень публикуемых CRAB AMQP-сообщений

| Наименование | Ключ маршрутизации | Описание |
|---|---|---|
| Нотификация о результатах обработки заявки | Ключ маршрутизации, указанный системой-клиентом при регистрации заявки | API продукта: response-модель нотификации о результатах обработки |

### 7.2.4. Перечень AMQP-сообщений, для которых CRAB является слушателем

| Наименование | Ключ маршрутизации | Публикатор | Описание |
|---|---|---|---|
| Регистрация заявок | Обрабатываются все сообщения из очереди, указанной в параметре компонента **CRAB_AKKA**: `ps.crab.crab_akka.rabbitmq.consuming.orders.new_orders.queue` | Система-клиент, регистрирующая заявку | AMQP API Регистрация заявок |
| Ответы по асинхронным взаимодействиям | Обрабатываются все сообщения из очереди, указанной в параметре компонента **CRAB_AKKA**: `ps.crab.crab_akka.rabbitmq.consuming.callbacks.oapi_callbacks.queue` | Система, с которой взаимодействует CRAB-сценарий | AMQP API Регистрация ответов по асинхронным взаимодействиям |

## 7.3. CRAB_AKKA REST API

| Группа интерфейсов | Описание функций |
|---|---|
| Заявки | OAPI Регистрация заявки<br>OAPI Регистрация команды на повторную обработку заявки<br>OAPI Отметка обработанной заявки как отмененной<br>OAPI Получение заявки<br>OAPI Получение списка заявок |
| Внешние события для сценария | OAPI Регистрация внешнего события<br>OAPI Получение внешних событий по заявке |

# 8. Архитектура развертывания и зависимости

Глава содержит сведения об архитектуре развертывания продукта, инфраструктуре для его установки, а также перечень продуктов Nexign, от которых зависит **CRAB**.

## 8.1. Системные требования

### Инфраструктура для установки CRAB

1. Операционная система RedOS.
2. JDK версии 1.8.
3. Контейнер сервлетов Apache Tomcat.
4. PostgreSQL.
5. «Сервер сообщений RabbitMQ» (RabbitMQ).
6. «WEB-сервер Apache» (APACHE).

## 8.2. Схема развертывания

Схема развёртывания **CRAB** представлена на Рис. 5.

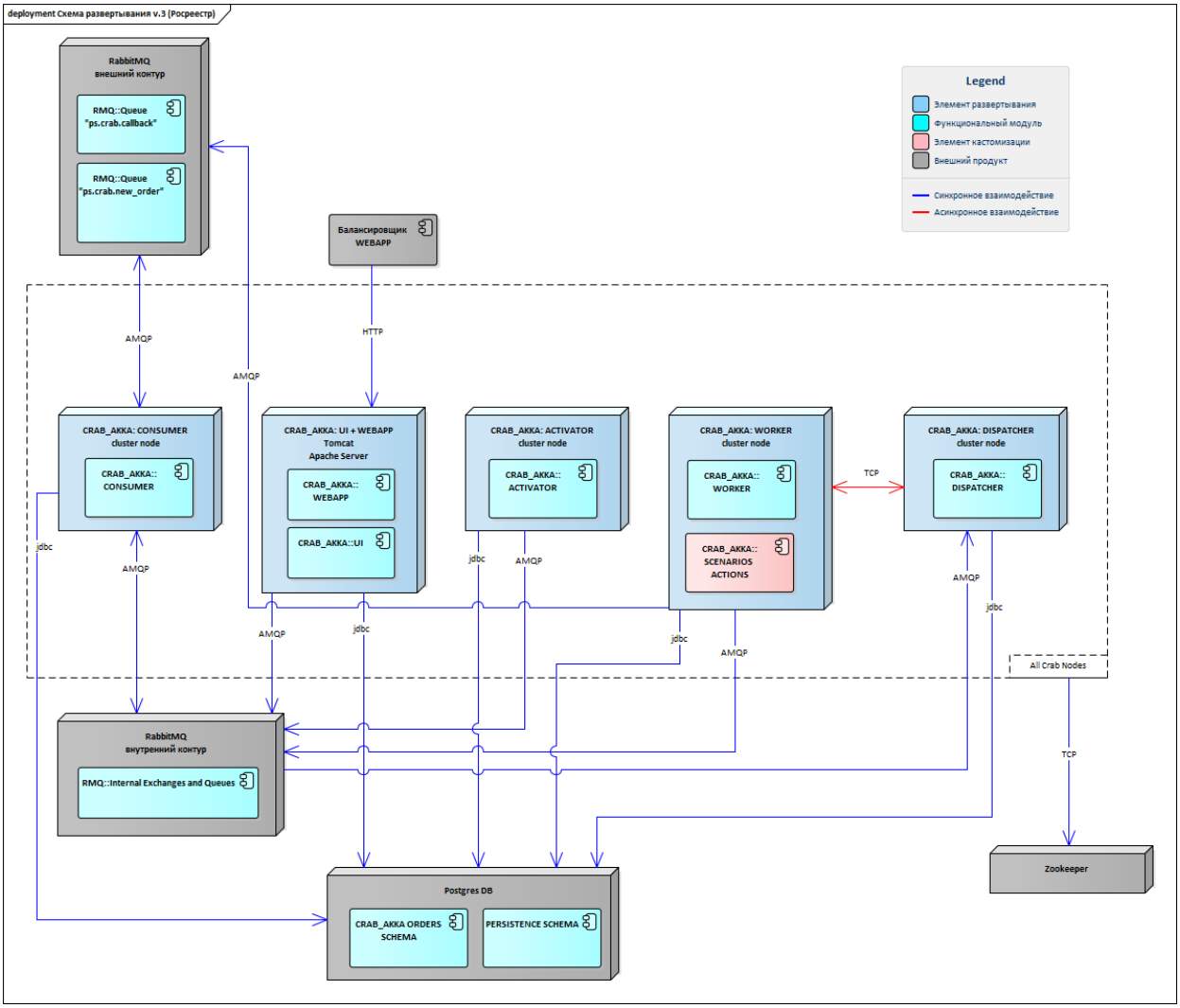


Рис. 5. Схема развёртывания **CRAB**

Описание элементов схемы развертывания **CRAB** и особенностей реализации приведено в Табл. 3.

Табл. 3. Элементы схемы развёртывания **CRAB**

| Элементы схемы развертывания | Описание | Особенности реализации |
|---|---|---|
| **CRAB_AKKA:UI** | Элемент развертывания для компонента UI. Содержит набор страниц пользовательского интерфейса | Реализован в виде отдельного web-приложения с набором статических html-страниц. Устанавливается в web-сервер Apache, развёрнутый на одном хосте с компонентом **WEBAPP**. Проксирует синхронные HTTP запросы пользователя на компонент **WEBAPP** того же хоста для получения данных, отображаемых в графическом интерфейсе. Для контроля доступа в web-интерфейс может быть настроено использование ldap |
| **CRAB_AKKA:WEBAPP** | Элемент развертывания для компонента **WEBAPP**. Обеспечивает работу REST-API продукта | Реализован в виде отдельного REST-сервиса на java. Устанавливается в контейнер сервлетов Tomcat. Взаимодействует:<br>• с хранилищем конфигурации при старте узла по протоколу TCP. Характер взаимодействия – синхронный;<br>• с внутренним брокером сообщений через протокол AMQP для публикации сообщений. Характер взаимодействия определяется настроечным параметром на уровне продукта. По умолчанию взаимодействие асинхронное (публикация сообщений без подтверждения);<br>• с системой хранения данных через протокол JDBC. Характер взаимодействия – синхронный |

| Элементы схемы развертывания | Описание | Особенности реализации |
|---|---|---|
| **CRAB_AKKA:DISPATCHER** | Элемент развертывания для компонента **DISPATCHER**. Обеспечивает работу функции распределения заявок | Реализован в виде JAVA-приложения. Для управления используется система **supervisord**. Является seed-узлом кластера обработки заявок. Представляет собой stateful-сервис. Состояние и его изменение хранится в базе данных, При запуске узла состояние восстанавливается. Взаимодействует:<br>• с узлом **WORKER** по протоколу TCP; характер взаимодействия – асинхронный;<br>• узлами, входящими в кластер **CRAB_AKKA** по протоколу gosip; характер взаимодействия – асинхронный;<br>• хранилищем конфигурации при старте узла по протоколу TCP; характер взаимодействия – синхронный;<br>• внутренним брокером сообщений через протокол AMQP для публикации сообщений; характер взаимодействия определяется настроечным параметром на уровне продукта; по умолчанию взаимодействие асинхронное (публикация сообщений без подтверждения);<br>• системой хранения данных через протокол JDBC; характер взаимодействия – синхронный |

| Элементы схемы развертывания | Описание | Особенности реализации |
|---|---|---|
| **CRAB_AKKA:WORKER** | Элемент развертывания для компонента **WORKER**. Обеспечивает работу функций обработки заявок и исполнения сценариев | Реализован в виде JAVA-приложения. Для управления используется система **supervisord**. Входит в кластер обработки заявок. Взаимодействует:<br>• с узлом **DISPATCHER** по протоколу TCP; характер взаимодействия – асинхронный;<br>• внешним брокером сообщений по протоколу AMQP для публикации сообщений; характер взаимодействия – асинхронный (публикация сообщений без подтверждения);<br>• внешними системами при исполнении сценария по протоколам HTTP и AMQP; характер взаимодействия – синхронный или асинхронный (зависит от особенностей функции, предоставляемой конкретной системой);<br>• узлами, входящими в кластер **CRAB_AKKA** по протоколу gosip; характер взаимодействия – асинхронный;<br>• хранилищем конфигурации при старте узла по протоколу TCP; характер взаимодействия – синхронный;<br>• внутренним брокером сообщений через протокол AMQP для обработки сообщений; характер взаимодействия – асинхронный;<br>• системой хранения данных через протокол JDBC; характер взаимодействия – синхронный |

| Элементы схемы развертывания | Описание | Особенности реализации |
|---|---|---|
| **CRAB_AKKA:CONSUMER** | Элемент развертывания для компонента **CONSUMER**. Обеспечивает работу AMQP-API продукта | Реализован в виде JAVA-приложения. Для управления используется система **supervisord**. Входит в кластер обработки заявок. Взаимодействует:<br>• с внешним брокером сообщений по протоколу AMQP для обработки сообщений; характер взаимодействия – синхронный;<br>• узлами, входящими в кластер **CRAB_AKKA** по протоколу *gosip*; характер взаимодействия – асинхронный;<br>• хранилищем конфигурации при старте узла по протоколу TCP; характер взаимодействия – синхронный;<br>• внутренним брокером сообщений через протокол AMQP для обработки сообщений; характер взаимодействия – асинхронный;<br>• системой хранения данных через протокол JDBC; характер взаимодействия – синхронный |

| Элементы схемы развертывания | Описание | Особенности реализации |
|---|---|---|
| **CRAB_AKKA:ACTIVATOR** | Элемент развертывания для компонента **ACTIVATOR**. Обеспечивает работу функции активации отложенных и приостановленных заявок | Реализован в виде JAVA-приложения. Для управления используется система **supervisord**. Входит в кластер обработки заявок. Взаимодействует:<br>• с узлами, входящими в кластер **CRAB_AKKA** по протоколу **gosip**; характер взаимодействия – асинхронный;<br>• хранилищем конфигурации при старте узла по протоколу TCP; характер взаимодействия – синхронный;<br>• внутренним брокером сообщений через протокол AMQP для обработки сообщений; характер взаимодействия – асинхронный;<br>• системой хранения данных через протокол JDBC; характер взаимодействия – синхронный |
| **ORDERS SCHEMA** | Система хранения информации о заявках | Представляет собой набор таблиц. Устанавливается на сервер DB |
| **PERSISTENCE SCHEMA** | Система хранения актуального состояния распределителя | Представляет собой набор таблиц. Устанавливается на сервер DB |