ЦИФРОВАЯ
ИНДУСТРИАЛЬНАЯ
ПЛАТФОРМА

Информация для разработчика

Zyfra Industrial Internet of Things Platform (ZIIoT) 2.20.0

Изменения в документе

Версия	Дата	Автор	Описание
1.0	14.08.2024	Пеплин Ф.Н.	Создание документа
1.1	25.11.2024	Кот О.В.	Обновление к релизу 2.20.0

Содержание

1. Введение

Изменения системы авторазвертывания (zifctl и инфраструктуры) и изменения в сервисах см. документе Сведения о релизе 2.20.0.

1.1. Переход на Cassandra 4

Начиная с версии платформы 2.20, по умолчанию будет устанавливаться 4 версия Cassandra. При обновлении предусмотрена автоматическая миграция данных. Новая версия Cassandra значительно улучшает производительность и стабильность базы данных, а также предоставляет доступ к расширенным функциональным возможностям.

Откат на версию 3 невозможен:

После перехода на версию 4 откат на версию 3 невозможен из-за изменений в структуре данных и механизмах работы самой базы данных. Это связано с тем, что Cassandra 4 использует новые внутренние форматы и алгоритмы, которые не совместимы с предыдущей версией.

Как вернуть версию 3:

Если по какой-то причине необходимо вернуть базу к версии 3, это можно сделать только через восстановление из резервной копии (бэкапа). Это означает, что до перехода на 4 версию вы должны создать актуальный бэкап данных. В случае необходимости этот бэкап будет единственным способом вернуть систему.

Как отказаться от обновления до версии 4:

Если вы хотите оставить 3 версию Cassandra, необходимо явно указать версию в конфигурационном файле `env-values.yaml`.

```
cassandra:
  version:                                                    3
  installed: True
```

Переход с увеличением количества реплик:

С переходом на Cassandra 4 вы можете увеличить количество реплик в кластере, что повышает отказоустойчивость и доступность данных.

```
cassandra:
  installed:                                                  True
  nodes: 3
```

Переход с версии Cassandra без включенного TLS на версию с активированным TLS:

Во время тестирования был выполнен переход с версии Cassandra без включенного TLS на версию с активированным TLS (Transport Layer Security). Основная цель тестирования заключалась в проверке работоспособности всех компонентов системы и стабильности соединений после включения TLS для обеспечения безопасной передачи данных.

```
cassandra:
  installed:                                                  True
  nodes:                                                         3
```

```
tls:
  enabled: True
```

Рекомендации:

Перед обновлением настоятельно рекомендуем создать резервную копию всех данных для предотвращения потерь в случае необходимости отката. Проверьте и протестируйте все критически важные функции приложения на версии 4 Cassandra перед развертыванием в продуктивной среде.

1.2. Apache Nifi

Внимание! При переключении режима авторизации из oauth-proxy в native, в соответствии с требованиями ИБ, происходит шифрование репозиториев NiFi. При этом создаётся специальный keystore и прописываются параметры nifi.repository.encryption.* в nifi.properties

Шифруются файлы в поде zif-nifi-0 (также 1,2 если включен high availability), контейнер server, директории: /opt/nifi/nifi-current/provenance_repository /opt/nifi/nifi-current/content_repository

Данный процесс не обратим! Переключение в автоматическом режиме с native в oauth-proxy не поддерживается в инсталляторе zifctl.

Возможно переключение NiFi с native в oauth-proxy вручную:

1. В nifi.properties вставить параметры, связанные с шифрованием репозиториев:

   nifi.repository.encryption.protocol.version=1 nifi.repository.encryption.key.id=nifi-repo-key
   nifi.repository.encryption.key.provider=KEYSTORE
   nifi.repository.encryption.key.provider.keystore.location=/opt/nifi/nifi-current/config-data/certs/repo_keystore.p12 nifi.repository.encryption.key.provider.keystore.password=

   Пароль для keystore можно взять из секретов развёртывания nifi: keystorePassword: (можно взять из лога развёртывания или вывода команды zifctl secrets show)

2. Взять, созданный при native режиме keystore, путь к файлу /opt/nifi/nifi-current/config-data/certs/repo_keystore.p12 . Скопировать его в файл /opt/nifi/nifi-current/config-data/certs/repo_keystore.p12 или примонтировать его в виде secret к поду zif-nifi-0 (1,2 при HA) в контейнер server по тому же пути.

3. Перезапустить NiFi

1.3. Доступ сервисов к нескольким БД в рамках одного модуля

В рамках zifctl реализована возможность получения доступа сервисом, даже не имеющего БД, к БД соседних сервисов в рамках одного модуля установки.
Для получения доступа необходимо до установки модуля разместить файл `service-list-patch.yaml` в директории services.

Рассмотрим пример.

Условия:

1. Сервис которому необходимо предоставить доступ - test-srv1
2. Сервис test-srv1 не имеет собственной БД
3. Пользователь для сервиса test-srv1 должен создать инсталлятор zifctl
4. Сервис к БД которого необходимо получить доступ test-srv2
5. Сервис к БД с нестандартным именем (`not_standart__db_name`) необходимо получить доступ
   Пример файла service-list-path.yaml

```
modules:
- name: test
  services:
   - name: test-srv1
     linkDb:
      - srvName: test-srv2
      - dbName: not_standart__db_name
        postgres: true
        postgresCreateDb: false
```

1.3.1. Примеры из практики

Вариант 1

Контекст:

У сервиса А есть своя БД и соответственно пользователь А.
Сервису А необходимо иметь доступ в базу сервиса Б под пользователем А.

Решение:

В файле `service-list-patch.yaml` добавляем необходимые строчки:

```
postgres: true
linkDb:
- srvName: zmeb-data-cache
```

В итоге блок будет выглядеть так:

```
- name: zmeb-user-activity #(Сервис А)
  image: zmeb-user-activity
  imageNamespace: zmeb
  tag: 4.11.1
  auth: confidential
  ingress: true
  ingressPath: "/zmeb-user-activity"
  ingressUrlRewrite: false
  type: netcore
  postgres: true
  linkDb:
   - srvName: zmeb-data-cache  #(Сервис Б)
     abac:
     enabled: false
     startupCheck: false
```

В `gotmpl` шаблон файла сервиса А `zmeb-user-activity.yaml.gotmpl` вставляем необходимые переменные:

```
environmentVars:
  POSTGRES_USER_DATA_CACHE:
    valueFrom:
      secretKeyRef:
        name:  "{{ .Values.ZifModule.chart }}-secrets"
        key: "zmeb-user-activity-psql-user"
  POSTGRES_PASSWORD_DATA_CACHE:
    valueFrom:
      secretKeyRef:
        name:  "{{ .Values.ZifModule.chart }}-secrets"
```

```
        key: "zmeb-user-activity-psql-pass"
  POSTGRES_DATABASE_DATA_CACHE:
    valueFrom:
      secretKeyRef:
        name:  "{{ .Values.ZifModule.chart }}-secrets"
        key: "zmeb-data-cache-psql-db"
  POSTGRES_HOST_DATA_CACHE: {{ .Values.postgres.host }}
  POSTGRES_PORT_DATA_CACHE: {{ .Values.postgres.port }}
```

Результат:

Сервис А может ходить в базу сервиса Б, потому что у пользователя БД сервиса А есть права на базу сервиса Б.

Вариант 2

Контекст:

У сервиса А нет своей БД и соответственно пользователя А.
Сервису А необходимо иметь доступ в базу сервиса Б и создать пользователя А.

Решение: В файле service-list-patch.yaml добавляем необходимые строчки:

```
...
postgres: true
postgresCreateDb: false
linkDb:
   - srvName: zmeb-data-cache
...
```

В итоге блок будет выглядеть так:

```
- name: zmeb-datacollector #(Сервис А)
  image: zmeb-datacollector
  imageNamespace: zmeb
  tag: 4.11.1
  auth: confidential
  ingress: true
  ingressPath: "/zmeb-datacollector"
  ingressUrlRewrite: false
  type: netcore
  postgres: true
  postgresCreateDb: false
  linkDb:
   - srvName: zmeb-data-cache #(Сервис Б)
     abac:
     enabled: false
     startupCheck: false
```

В `gotmpl` шаблон файла сервиса А `zmeb-user-activity.yaml.gotmpl` вставляем необходимые переменные:

```
environmentVars:
  POSTGRES_USER_DATA_CACHE:
    valueFrom:
      secretKeyRef:
        name:  "{{ .Values.ZifModule.chart }}-secrets"
```

```
      key: "zmeb-datacollector-psql-user"
POSTGRES_PASSWORD_DATA_CACHE:
  valueFrom:
    secretKeyRef:
      name:  "{{ .Values.ZifModule.chart }}-secrets"
      key: "zmeb-datacollector-pass"
POSTGRES_DATABASE_DATA_CACHE:
  valueFrom:
    secretKeyRef:
      name:  "{{ .Values.ZifModule.chart }}-secrets"
      key: "zmeb-data-cache-psql-db"
POSTGRES_HOST_DATA_CACHE: {{ .Values.postgres.host }}
POSTGRES_PORT_DATA_CACHE: {{ .Values.postgres.port }}
```

Результат:

Создан пользователь А, без базы для сервиса А. Пользователь А может ходить в базу сервиса Б, потому что у пользователя БД сервиса А есть права на базу сервиса Б.

2. Требования по проектированию приложений на платформе

2.1. Модальные глаголы

1. Необходимо (MUST), а также требуется (REQUIRED) и должно (SHALL) — абсолютное требование.
2. Недопустимо (MUST NOT), а также не должно, не допускается (SHALL NOT) — абсолютный запрет.
3. Следует (SHOULD), а также рекомендуется (RECOMMENDED) — требования, от которых можно отказаться при наличии разумных причин и обоснований.
4. Не следует (SHOULD NOT), а также не рекомендуется (NOT RECOMMENDED) — требования, нарушение которых возможно при наличии обоснований, но может вызвать проблемы.
5. Допускается, могут, возможно (MAY), а также необязательный (OPTIONAL) — возможности, предоставляемые платформой, использование которых допустимо, но не обязательно.

2.2. Требования

При редактировании документа новые требования должны добавляться в конец списков, чтобы не нарушать нумерацию требований. Требования могут удаляться.

2.2.1. Моделирование

1. Все данные приложений, к которым необходим доступ другим приложениям, должны быть включены в Объектную Модель и доступны другим приложениям через сервисы OM/UDL.
2. Приложения должны использовать Единую Объектную Модель (EOM), описывающую ресурсы и деятельность всего предприятия.
3. Следует проектировать разделяемые между приложениями модели EOM. Допускается разделение данных EOM на несколько областей применения, приложения могут создавать собственные области применения, если существующие не позволяют описать предметную область.
4. Приложениям следует использовать семантические сервисы OM, реализующие модели ISA-95, при автоматизации соответствующих процессов.
5. Взаимодействие с внешними относительно платформы системами (например, 1C, SAP) следует проектировать путем создания в платформе процессов переноса данных в/из внешних систем на базе Apache NiFi или других коннекторов к системе сбора данных платформы.
6. Следует рассмотреть перенос максимум данных из внешних источников в платформенные хранилища.
7. Возможно подключение внешних источников данных (SQL, Hive, GE Historian etc) через средства DataReference UDL/OM.
8. Не допускается хранить логику или расчеты приложения во внешних источниках данных.
9. Приложениям следует выполнять миграции данных в сервисах OM в соответствии с документом ADR Миграция данных приложений в объектную модель.

10. Для моделирования документов, их структуры, набора полей, связи с S3 хранилищем рекомендуется использовать сервис документов (zif-document-archive).
11. При моделировании объектов, выбор источника данных для свойств следует производить на основе сравнительного анализа гибкости конфигурации и производительности выборки данных.
12. При использовании источника данных Константа, сле дует принимать во внимание, что ее значение хранится в конфигурации свойства ОМ, что усложняет процесс переноса конфигураций ОМ между стендами. Константы рекомендуется использовать для свойств, значения которых не меняются с течением времени.
13. Приложения могут создавать адаптационные модели для упощения выборки/агрегации данных из общих моделей EOM.

2.2.2. Хранение данных

1. Данные, для хранения которых в платформе есть специализированные сервисы (ОМ, БДВР, семантические модели ISA-95) следует хранить в соответствующих сервисах.
2. Временные ряды должны храниться в сервисах БДВР платформы.
3. Кэширование следует реализовывать с использованием Redis.
4. Приложениям не следует разворачивать самостоятельно сервисы, аналогичные предоставляемым платформой:
   - Реляционная БД — приложения могут использовать платформенную PostgreSQL.
   - БДВР — приложения могут использовать платформенную БДВР для хранения временных рядов связанных со свойствами ОМ.
   - Брокер сообщений — приложения могут использовать платформенный Apache Kafka\Schema registry, RabbitMq.
   - Неструктурированные данные/файлы — приложения могут использовать платформенное S3-совместимом хранилище доступное через s3-proxy сервис zif-file-storage.
5. Приложениям следует использовать сервисы UDL для получения WebSocket подписки на изменения значений свойств ОМ.
6. Приложения могут использовать шаблон издатель-подписчик для обмена данными с другими приложениями или сервисами платформы через брокеров сообщений Apache Kafka\RabbitMq:
   - Apache Kafka будет предпочтительней в случаях, когда требуется большая пропускная способность/потоковая обработка передаваемых данных.
   - RabbitMq будет предпочтительней в сценариях, требующих маршрутизации/точечной доставки сообщений контрактным подписчикам.
7. Для формирования витрин данных по моделям EOM/значениям свойств рекомендуется использовать сервис универсальной витрины данных — zif-universal-datamart.

2.2.3. Доступ к данным

1. Приложения должны обращаться к данным через сервисы ОМ/UDL/Events, либо через их публичные шины сообщений.
2. Сервисы не должны напрямую обращаться к платформенной СУБД Cassandra.

3. Сервисы не должны обращаться к тегам БДВР по имени, все обращения к данным должны быть через свойства ОМ и сервисы UDL.
4. Приложения не должны напрямую подключаться к внешним источникам данных.
5. Приложению должно получать доступ к справочникам НСИ через zif-rdm-common (ранее zif-om-directories) не зависимо от того, является ли мастер-системой для конкретного справочника платформа ZIIot или внешний источник, данные из которого переносятся в zif-rdm-common.
6. Не допускается использование Apache NiFi для формирования бизнес логики приложений, выходящей за рамки ETL процессов или в случаях наличия специализированных сервисов.
7. В случае необходимости в взаимодействиях с внешними приложениями, которые по обоснованным причинам не могут быть реализованы в качестве приложений на платформе, рекомендуется разрабатывать отдельные сервисы-адаптеры.
8. В случае необходимости в взаимодействиях с внешними источниками данных, которые по обоснованным причинам не могут быть перенесены в платформу, рекомендуется разрабатывать отдельные сервисы-адаптеры.

2.2.4. Рабочие процессы, события и уведомления

1. При разработке бизнес-логики приложений, в том числе с участием пользователей, следует использовать нотацию BPMN и сервис рабочих процессов, обеспечивающих ее выполнение.
2. Производственные события, если они требуют регистрации, используются для интеграции с рабочими процессов/уведомлениями или другими решениями, должны храниться в сервисе событий (zif-events).
3. Следует использовать сервис уведомлений для рассылки уведомления о зарегистрированных событиях через выбранные каналы.
4. Следует использовать сервис уведомлений для рассылки уведомления о событиях внешних систем не зарегистрированных в сервисе событий.
5. Следует минимизировать создание скриптовых обработок внутри BPMN процессов и по возможности всегда использовать паттерн внешних задач для обработки и взаимодействия с данными (сравнение подходов).
6. Создание скриптовых обработок внутри BPMN допустимо в случаях:
   - отсутствия необходимых внешних обработчиков и невозможности их реализовать;
   - необходимости выполнить запрос с минимально возможной задержкой и небольшой длительностью выполнения.
7. Для взаимодействия с данными ZIIoT из BPMN процессов следует использовать внешние задачи (external tasks) исполняемые в сервисах zif-bp-*.

2.2.5. Расчеты

1. При необходимости расчетов следует использовать свойства объектной модели с источником данных Формула (calctag).
2. Для расчета формул из ОМ не следует использовать сервис потоковых расчетов zif-cm-xxx напрямую, в обход ОМ.
3. Допускается для генерации событий по спецификациям проверок использовать решения EventGenUdl на базе Apache Spark.

4. Допускается использовать кастомные расчеты на базе Apache Spark в тех случаях, когда функционал невозможно реализовать на сервисе потоковых расчетов.

2.2.6. Инфраструктура

1. Приложение должно быть рассчитано на работу внутри кластера платформы (Kubernetes 1.19+/OpenShift 4.0+) и не требовать для размещения отдельных виртуальных машин.
2. Приложение не должно требовать создания кастомных CRD или требовать предоставления прав уровня кластера для своего развертывания или функционирования.
3. Сервисы, составляющие приложение, должны поставляться как docker-образы на базе ОС Linux,
4. Докер образы сервисов должны быть собраны с учетом запуска от имени непривелигированного пользователя со случайным UID (см. требования по созданию docker-образов для OpenShift).
5. У сервисов должна быть версия в формате SemVer2.
6. Сервисы приложений, доступные снаружи кластера (например, доступные из фронтенд) должны быть опубликованы только стандартным для Платформы образом с использованием Ingress-контроллера (Route-контроллера для OpenShift) кластера.
7. Не допускается делать доступным снаружи кластера сервисы приложений через незащищенный HTTP.
8. Приложение не должно зависеть от каких-либо сервисов, кроме платформенных или сервисов приложения, развернутых внутри кластера. Не допускается развертывать зависимости приложения в отдельных виртуальных машинах.

2.2.7. Разработка, конфигурация и запуск сервисов

1. В большинстве случае рекомендуется создавать stateless сервисы, допустимо использование stateful-сервисов, в сценариях, когда наличие состояние значительно повышает эффективность решения, сервисы должны обеспечивать сохранение состояния в соответствующих Платформенных хранилищах (ОМ, PostgreSQL, Kafka, S3 storage и т. д.) и не терять данные во время перезапуска.
2. Сервисы должны получать настройки через переменные окружения контейнеров.
3. Если приложению необходимы для старта какие-то данные или конфигурацию в любых постоянных хранилищах, то приложение должно самостоятельно создать всю начальную конфигурацию в размере, достаточном для запуска и дальнейшей конфигурации средствами приложения.
4. Возможно требовать от пользователя/администратора после первого старта приложения дополнения конфигурации средствами приложения.
5. Для всех настроек, для которых это осмысленно, следует иметь значения, по умолчанию зашитые внутрь контейнера. Этим значениям рекомендуется подходить для большинства применений.
6. Настройки, зависящие от окружения, не должны зашиваться (в т.ч. IP-адреса, ссылки на домены, включая localhost, имена серверов СУБД, и тому подобное) и не должны иметь значения по умолчанию.
7. Сервисы должны поддерживать горизонтальное масштабирование, с балансировкой средствами сервисов Kubernetes в случае использования простых алгоритмов балансировки таких как round-robin, допускается использования собственных сервисов балансировки, в случаях, когда требуются специализированные алгоритмы, не поддерживаемые в Kubernetes.

8. Сервисы должны обрабатывать ситуацию временной недоступности зависимостей, ожидать и возобновлять работу при восстановлении их доступности (состояние работы сервиса должно отражаться в readiness-пробе, см. ниже).
9. Сервис, которому не заданы обязательные параметры при старте, должен сам завершить свою работу. В этом случае сервису следует выдать в лог детальное описание ошибки.
10. Если на старте сервиса он не смог подключиться к необходимым зависимостям (например, к БД), он должен либо самостоятельно завершить работу с выдачей на консоль детального описания ошибки, либо перейти в цикл попыток подключения к инфраструктуре и продолжить начальную инициализацию после появления доступа.
11. Сервис должен запускаться обладая только минимально необходимыми правами на инфраструктурные зависимости. Например, сервис не должен ожидать наличия прав на создание базы данных у той учетной записи, которая ему передана.
12. Переменные окружения, связанные с сервисами платформы (например, указывающий на инфраструктурные сервисы или сервисы платформы) следует иметь стандартные имена согласно документу Именование переменных. POSTGRES_XXX, AUTH_XXX, REST_XXX и т.д. Это упростит развертывание приложения, т.к. можно использовать автоматически создаваемые ConfigMap Платформы для задания значений этих переменных, и облегчит в дальнейшем переход на платформенные средства развертывания через маркетплейс.
13. Приложения должны обеспечить работоспособность при откате версии сервисов на предыдущую, без отката схемы данных. При разработке приложения необходимо соблюдать обратную совместимость схемы хранения данных в рамках минорных версий сервисов.
14. Сервисам следует использовать s3 хранилище для хранения файлов с структурированными и неструктурированными данными.
15. Сервисы не должны использовать файловую систему узла для хранения разделяемых между экземплярами сервиса данных.
16. Сервисы не должны предъявлять требования на использование persistent volumes с режимом ReadWriteMany при развёртывании в среде, так как используемый class storage может не поддерживать данный режим.
17. Корневая файловая система контейнера должна монтироваться в режиме только чтение. Для контейнера должны быть явно выделены ресурсы для записи через Persistent Volumes Claims.
18. Состав пробрасываемых в контейнер портов должен быть минимально достаточен для работы. Неиспользуемые в ходе работы контейнеризированного приложения порты не должны пробрасываться в контейнер.
19. Каждый процесс приложения должен быть запущен в отдельном контейнере, связанные контейнеры должны быть объединены в один Pod.

2.2.8. Мониторинг и наблюдаемость

1. Сервис должен предоставлять стандартные health-check пробы типов liveness и readiness по URL-путям /health/liveness и /health/readiness соответственно. Подробнее о семантике проб см. документацию kubernetes и требованиях по реализации проб.
2. Сервис может предоставлять пробу типа startup по URL /health/startup.
3. Сервис должен иметь endpoint для отдачи метрик в формате Prometheus по адресу /metrics.
4. Сервис должен отдавать базовые метрики (для ASP.NET Core и runtime или общий пакет) о состоянии рантайма (например, объем потребляемой памяти и т.д.).

5. Сервису следует отдавать специфические для сервиса метрики (например, количество вычитанных и обработанных событий).
6. Сервис должен выдавать свой лог на консоль, рекомендации по логированию описаны в документе Технические рекомендации по реализации логирования в сервисах.
7. Сервису следует логировать все ключевые события (в том числе обработку запроса, разрешение доступа, отказ в доступе)
8. Сервису следует иметь возможность включения/отключения детализированного (отладочного) логирования через переменную окружения.
9. Сервисы должны поддерживать трассировку в формате OpenTracing (Jaeger).
10. Сервисы, содержащие HTTP API, должны предоставлять описание своего API в формате OpenAPI/Swagger.

2.2.9. Безопасность

1. Сервисы должны обеспечивать аутентификацию при любом обращении к API или фронтенду и только с помощью протокола OpenID Connect, включая аутентификацию при взаимодействиях между бекенд-сервисами (справочно: Технические рекомендации по безопасности при микросервисной архитектуре).
2. Frontend-сервисы, встроенные в портал, должны получать токен авторизации от портала.
3. Frontend-сервисы, не встроенные в портал или встроенные через IFrame, должны использовать процедуру OpenID Connect Authorization Code Flow для аутентификации пользователей.
4. Фронтенд-сервисы должны передавать полученный токен пользователя в любых запросах к бэкенд-сервисам.
5. Бэкенд-сервисы должны соблюдать принцип identity propagation (вызов всех http сервисов в цепочке обработки запроса должен осуществляться от имени исходного пользователя) – передавая токен входящего http-запроса во все исходящие запросы, совершенные в рамках конвейера обработки запроса. Токен входящего запроса также должен быть использован и для случаев, когда часть обработки идет асинхронно вне конвейера http-запроса, в этом случае токен рекомендуется передавать обработчику в виде части сообщения.
6. В случае, если сервис выполняет функциональность общего назначения, не связанный с действиями конкретного пользователя, то вызовы к другим сервиса следует производить от имени клиента сервиса (oauth client) используя поток получения токена доступа *OpenID Connect Client Credentials Flow (например, это могут быть фоновые расчеты или прогрев данных кэша и т.д.).
7. При реализации авторизации по любой модели, сервисы не должны использовать дополнительных Claim внутри JWT-токена, помимо стандартных для JSON Web Token, roles и user atttributes.
8. Приложения не должны ожидать, что KeyCloak будет подключен к конкретному провайдер аутентификации (например, именно к Active Directory).
9. Приложениям рекомендуется использовать авторизация по модели ABAC (Attribute-Based Access Control) с использованием платформенного сервиса авторизации (zif-security).
10. Приложениям не рекомендуется (но возможно) использовать авторизацию по модели RBAC (Role-Base Access Control) на базе собственной ролевой модели приложения, основываясь на данных из JWT-токена.
11. При выборе модели авторизации следует руководствоваться ADR Требования к аутентификации, авторизации и аудиту сервисов.

12. Сервисы должны обеспечивать возможность использования сертификата TLS для включения HTTPS соединения.
13. Докер образы приложений не должны содержать зарегистрированных CVE уязвимостей (Common Vulnerabilities and Exposures) уровня High и выше и строиться на актуальных версиях операционных систем и пакетов приложений.
14. Не допускается пробрасывать привилегированные порты (до 1024) в контейнер.
15. Не допускается наличие в образе контейнера файлов с полномочиями setuid, setgid.
16. Не допускается устанавливать и запускать ssh-сервер и прочие средства удаленного управления в контейнере.
17. Контейнер приложения должен быть максимально ограничен в части использования Linux Capabilities (например, обычно не требуются возможности NET_ADMIN, SYS_ADMIN, SYS_MODULE), а также не должен выходить за рамки Seccomp-профиля. Исключения прорабатываются отдельно для каждого приложения.

2.2.10. UI

1. Приложения, разработанные или адаптированные к платформе, должны быть встроены в портал платформы.
2. Приложения, разработанные на платформе, должны подключаться в портал платформы с использованием библиотеки single-spa.js, а адаптированным — следует это сделать.
3. Приложения, адаптированные к платформе и интегрированные с ней, могут работать через iframe-встраивание.
4. Приложения рекомендуется реализовывать с использованием фреймворка Angular или библиотек React/Vue, т.к. подключение их в настоящий момент протестировано и поддерживается.
5. Приложениям рекомендуется следовать требованиям Guideline ZIIoT — рекомендации по проектированию пользовательских интерфейсов.
6. Использование дизайн-системы для ZIIoT:
   - Проектируемым приложениям рекомендуется использовать дизайн-систему Prizm (версии 4.0 - Prizm это развитие дизайн-системы ZIIoT версии 3.0).
   - Разрабатываемым приложениям рекомендуется запланировать переход на дизайн-систему Prizm.
   - Если приложение не использовало компонентную базу, то рекомендуется использовать дизайн-систему Prizm.
7. Если приложение использовало (начало миграцию) компонентную базу версию 3.0, то рекомендуется продолжить ее использование (продолжить миграцию), до разработки мигратора до Prizm.
   - Проектам, разработанным без компонентной базы, рекомендуется запланировать переход сразу на дизайн-систему Prizm, если планируется продолжать разработку приложения год или более.
   - Рекомендуется использовать графики из Prizm для отображения визуализации в приложениях.
8. Приложениям рекомендуется использовать платформенные средства визуализации (отчеты, дашборды, мнемосхемы) для отображения данных из ОМ, что позволит максимально гибко

отображать данные (в т.ч. показывать на экранах приложений данные из других приложений).

9. Приложения должны работать с оболочкой портала только с использованием средств предоставленных SDK.
10. Приложения должны инкапсулировать собственные стили. Не допускается влиять или создавать стили на уровне глобальной области видимости. Необходимо очищать свои стили при разрушении приложения.
11. Приложениям следует работать в своем окружении. Не следует определять в globalThis, window, document.
12. Приложения, разработанные на платформе, для взаимодействия с функционалом портала должны использовать сервис оболочку портала (zui-app-shell) .
13. Приложениям следует использовать сервис Сервис настроек приложения (zif-portal-settings-dotnet) для хранения настроек/состояния приложения, компонентов, сетки.
14. Приложения не должны содержать UI в виде нативных десктоп приложений.
15. Приложения должны использовать собственные ресурсы, не зависеть от ресурсов AppShell (assets, css).
16. При разработке приложений рекомендуется поддерживать все браузеры, попадающие под следующие условия (согласно browserslist) и, в случае использования, Prizm:
    - распространенность более 0.5% на глобальном WWW;
    - последние 2 версии каждого браузера;
    - FireFox ESR;
    - есть поддержка;
    - не IE 9-11.

2.2.11. Представление архитектуры приложения

Рекомендуется представлять архитектуру приложений в соответствии с документом Соглашение по моделированию архитектуры.

2.3. Уровни требований

При проектировании приложения уровень его интеграции с платформой ZIIot устанавливается при утверждении плана проекта. Уровни интеграции:

1. Приложение, разрабатываемое под платформу ZIIot, то есть приложение, при проектировании которого изначально закладывалась необходимость работы на платформе ZIIot.
2. Приложение, адаптированное (смигрированное) под платформу ZIIot, то есть приложение, изначально при проектировании которого не закладывалась необходимость работы на платформой ZIIot или закладывалась не в полном объеме, но потом было принято решение об необходимости работы на платформе ZIIot.
3. Приложение, интегрированное с платформой ZIIot, то есть приложение, которое разрабатывается для работы без платформы ZIIot, но имеется необходимость заложить опциональную возможность работы с платформой ZIIot.

2.3.1. Приложение, разрабатываемое под платформу ZIIoT

Приложение, разрабатываемое под платформу ZIIot, проектируется и реализуется, максимально используя все возможности платформы.

Отступление от обязательных требований (должно) не допускается. В случае, если решением архитектурного комитета ZIIoT установлена такая необходимость, то инициируется процесс пересмотра или дополнения обязательных требований.

Отступление от рекомендуемых требований (следует, рекомендуется) допускается, только если необходимость такого отступления обоснована и доказана архитектурным проектированием приложения, и выполнение рекомендаций приведет к невыполнению (не полному выполнению) приложением функциональных или нефункциональных требований. Ссылка на то, что выполнение требования приведет к необходимости существенной переработки архитектуры приложений или уже реализованных программных модулей не допускается.

Необходимость подтверждается на защите на архитектурном комитете ZIIoT.

2.3.2. Приложение, адаптированное (смигрированное) под платформу ZIIoT

При адаптации существующего приложения должен быть разработан поэтапный план миграции на целевую архитектуру. Целевая и промежуточная архитектура согласуется с архитектурным комитетом ZIIoT.

Промежуточная архитектура:

В промежуточной архитектуре представляется и защищается на архитектурном комитете ZIIoT список требований к приложениям, которые временно не соблюдается. Как правило, приложениям не следует в промежуточной архитектуре отступать от обязательных требований (должно). В частности, не допускается нарушения требований:

- Интеграция с другими приложениями через Объектной Модели/UDL (1.1).
- Работа приложения в портального решения платформы (10.1).
- Разворачивание приложения (6.1–6.8).
- Аутентификация (9.1–9.8).
- Не исполнение других требований, если нарушение их приведет или может привести к нестабильности работы платформы ZIIoT или других приложений.

Целевая архитектура:

Отступление от обязательных требований (должно) в целевой архитектуре не допускается.

Отступление от рекомендуемых требований (следует, рекомендуется) в целевой архитектуре допускается если выполнение рекомендуемых требований нецелесообразно (в том числе по причине необходимости существенной переработки приложения без получения существенной выгоды для функциональных или нефункциональных требований).

В частности, разрешается сохранять и не перерабатывать уже реализованное:

- Хранение и доступ к данным в выделенной под приложение БД PosgreSQL.
- Бизнес-логику, логика расчетов, классификации, обработки событий и др., даже при наличии в платформе аналогичных механизмов.
- Средства визуализации и ввода данных.
- Ролевая модель приложения, авторизация пользователей (но не аутентификация).

2.3.3. Приложение, интегрированное с платформой ZIIoT

При интеграции с платформой со стороны приложения предъявляются требования:

- Интеграция с системой аутентификации платформы.
- Все данные, которые потенциально требуются другим системам, должны быть доступны через `Объектную Модель/UDL` (1.1).
- Внешние взаимодействия могут быть реализованы через сервисы адаптеры.

Все остальные сервисы платформы используются по желанию. Приложение хостится вне кластера платформы и пользуется только сервисами платформы, доступными через `gateway`.

2.3.4. Требования для inline приложений

Приложения с типом **inline** реализованы через механизм iframe в браузере. В связи с этим есть ряд ограничений, связанный с безопасностью этого встраивания. **Это важно, только в тех случаях, если CSP уже используется на встраиваемом веб-сайте.**

Политика безопасности содержимого (Content Security Policy, CSP) управляется сервером, на котором размещается веб-контент. Эта политика объявляется в HTTP-заголовках и определяет, какие ресурсы клиенту (т. е. веб-браузеру) разрешено загружать с данного сервера. Правила служат дополнительным уровнем защиты от атак межсайтового скриптинга, инъекций данных и других подобных атак.

Чтобы сайт эффективно загружал ресурсами и скриптами, он должен:

- Включить и настроить CSP: Обычно это делается путем добавления HTTP-заголовка «Content-Security-Policy» с соответствующими директивами. Альтернативный подход включает установку тега в HTML.

Пример:

- Добавление header Content-Security-Policy: default-src 'self'; img-src https://*; child-src 'none';
- Добавление <meta http-equiv="Content-Security-Policy" content="default-src 'self'; img-src https://*; child-src 'none';">

- Определите «Источники» в политиках сервера: Разработчик должен перечислить список источников, которые безопасны для получения и выполнения. Например, script-src 'self' <ссылка> позволит загружать скрипты только из того же источника и <ссылка>.

Пример: `Content-Security-Policy: default-src 'self'; script-src 'self' <ссылка>; style-src 'self' 'unsafe-inline' <ссылка>`

1. Данный header является ключевым при встраивании в iframe. Использование frame-ancestors в политике: Чтобы разрешить другим сайтам вставлять его содержимое, сайт должен указать в своей политике frame-ancestors 'self' <ссылка>. Если сайт хочет разрешить всем источникам вставлять свое содержимое, ему следует указать frame-ancestors *. Данное свойство нельзя установить через , а только в header ответа.

2. Эта директива контролирует, из каких источников можно встраивать документ. Таким образом, она эффективно защищает от атак типа «кликджекинг», ограничивая круг лиц, которые могут встраивать ваш контент.

3. Она заменяет старые HTTP-заголовки X-Frame-Options. Если на сайте установлена директива frame-ancestors, пользовательский агент будет игнорировать любые заголовки X-Frame-Options.

4. Директива frame-ancestors поддерживает те же значения, что и другие директивы CSP ('self', 'none', data:, domain.com и т. д.), за исключением 'unsafe-inline' и 'unsafe-eval'.

Пример: `Content-Security-Policy: frame-ancestors 'self' <ссылка>;`

Отключите X-Frame-Options: CSP - это новая, более мощная альтернатива X-Frame-Options. Если CSP внедрен, рекомендуется удалить или отключить X-Frame-Options.

3. Контакты технической поддержки

Таблица 3.1. Контакты технической поддержки

Вид поддержки	Значение
Портал	https://jira.zyfra.com/servicedesk
Email	dp-support@zyfra.com

Регистрация запросов производится круглосуточно. Рабочее время — с 8.00 до 17.00 (время московское), перерыв с 12.00 до 13.00, в рабочие дни. Запросы, поступившие в нерабочее время, обрабатываются на следующий рабочий день.

Таблица 3.2. Контакты технической поддержки ЦИП

Вид поддержки	Значение
Портал	https://jira.zyfra.com/servicedesk/customer/portal/42
Email	support@idpllc.ru

Регистрация запросов производится круглосуточно. Рабочее время — с 9.00 до 18.00 (время московское), перерыв с 13.00 до 14.00, в рабочие дни. Запросы, поступившие в нерабочее время, обрабатываются на следующий рабочий день.