ПРОГРАММНЫЙ КОМПЛЕКС

ПЛАТЕЖНЫЙ ШЛЮЗ

**Руководство системного программиста**

**Листов \_15**

Инв. № подл.

Подп. и дата

Взам. инв. №

Инв. № дубл.

Подп. и дата

2021

**АННОТАЦИЯ**

Документ содержит описание структуры, процедуры установки, и проверки программного комплекса платежный шлюз и предназначен для системных программистов, осуществляющих подготовку программы к эксплуатации.

Настоящий документ предназначен для персонала, имеющего квалификацию системного администратора, обладающего знаниями и навыками в области системного администрирования, в том числе установки ОС Linux Debian, работы с СУБД MySQL, установки и настройки RabbitMQ.

Документ не является руководством пользователя по установке и настройке, операционной системы, и прочего системного и технологического программного обеспечения.

**ОГЛАВЛЕНИЕ**

[1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ 4](#_Toc91663372)

[2. СТРУКТУРА ПРОГРАММЫ 5](#_Toc91663373)

[3. УСТАНОВКА ПРОГРАММЫ 7](#_Toc91663374)

[3.1. Порядок установки 7](#_Toc91663375)

[3.1.1. Установка базы данных 7](#_Toc91663376)

[3.1.2. Установка RabbitMQ 8](#_Toc91663377)

[3.1.3. Настройка Kubernetes 8](#_Toc91663378)

[3.1.4. Установка сервисов в кластере Kubernetes 9](#_Toc91663379)

[4. ПРОВЕРКА ПРОГРАММЫ 12](#_Toc91663380)

[5. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ 13](#_Toc91663381)

[6. ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 14](#_Toc91663382)

# 1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

Программный комплекс платежный шлюз представляет собой набор программных сервисов, расположенных на удалённом сервере. Структурно платежный шлюз состоит из двух основных частей – **Frontend** и **Backend**.

**Frontend** представляет собой визуальную часть, отвечающую за взаимодействие с пользователем (видимая часть сайта в браузере компьютера или смартфона).

**Backend** отвечает за реализацию рабочих алгоритмов платежного шлюза и взаимодействие с другими сервисами.

Состав программы приведен в п. 2 настоящего документа.

Для обеспечения выполнения программы необходимы следующие технические средства:

Аппаратно-программные ресурсы выделенного сервера:

* предустановленная ОС Linux Debian версии 9;
* СУБД: MySQL, Redis.
* Развёрнутый Kubernetes-кластер.
* Сервис очередей RabbitMQ.

# 2. СТРУКТУРА ПРОГРАММЫ

Программный комплекс платежный шлюз представляет собой набор программных сервисов:

**ServiceGate –** точка входа запросов от сервисов и проектов. Выполняет задачи по приему запросов в соответствии со схемой GraphQL, передает их в другие сервисы платежного шлюза и возвращает ответ сервису/продукту.

**PaymentsReview** – сверка платежей сверяет платежи из реестра платежного агента и платежи в платежном шлюзе, находит расхождения и отправляет отчет по сверке. Сверка запускается на конкретную дату и отдельно по каждому платежному агенту. Запуск настроен по расписанию.

**CreateOrder** – осуществляет бизнес логику сценария получения и передачи ссылки на оплату. Выполняет задачи:

* Принимает от ServiceGate запросы на генерацию счета;
* Отправляет в биллинг проекта запрос на проверку аккаунта;
* Отправляет запрос на генерацию ссылки в адаптер платежного агента и возвращает в ServiceGate;
* Создает в базе данных заказ order.

**AcceptPayment** – прием платежей. Получает платежи через микро сервис-адаптеры платежных операторов/агентов, создание задание на отбивание кассового чека (если нужен). Передает информацию о платежах в биллинг, через микро сервисы-адаптеры биллингов. Контролирует дублирование оплат и разнесение неправильных платежей путем изменения получателя.

**Cashier** – принимает запрос на отправку в онлайн-кассу от AcceptPayment, проверяет переданные параметры. На основании переданных параметров определяет кассовый аппарат и оператора кассы. Сохраняет данные чека в БД. Отправляет чек в Атол с обратным ответом от Атол о статусе чека. Самостоятельно запрашивает статус отправки чека клиенту, обновляет статус отправки.

**Адаптер платежного агента** – является точной входа от платежного агента. Выполняет задачи:

* Проверяет параметры запроса и приводит данные к единому формату. Проверяемые параметры: тип команды, сумма, наличие обязательных параметров, ключи. Перечень проверок зависит от платежного агента.
* Преобразует данные для ответа в нужном для платежного агента формате.

**RefundPayment** – возврат/отмена платежа. Осуществляет бизнес-логику сценария возврата/отмены платежа. Выполняет задачи:

* Проверяет наличие платежа, Сохраняет возврат в базе данных платежного шлюза;
* Если запрос не от платежного агента, то запрашивает через адаптер платежного агента разрешение на возврат;
* Отправляет запрос на возвратный чек.

# 3. УСТАНОВКА ПРОГРАММЫ

Перед развёртыванием компонентов платежного шлюза, на сервере должна быть предустановлена ОС Debian версии 9. Компоненты платежного шлюза могут быть установлены как на одном сервере, так и каждой подсистеме может быть выделен собственный сервер/виртуальная машина.

## 3.1. Порядок установки

## 3.1.1. Установка базы данных

Для функционирования сервисов из состава программы необходимо установить СУБД MySQL. Для этого необходимо произвести установку пакета:

**apt install mysql-server**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**mysql -u root**

**mysql>**

**CREATE DATABASE paymentgate $**

**CREATE USER ' имя '@'%' IDENTIFIED BY 'пароль';**

**GRANT ALL PRIVILEGES ON \* . \* TO ' paymentgate '@'%';**

**FLUSH PRIVILEGES.**

В полях «имя» и «пароль» прописать имя пользователя и пароль.

Для функционирования сервисов Cashier и ServiceGate необходимо установить Redis:

**apt** install redis-server

## 3.1.2. Установка RabbitMQ

Для функционирования сервиса AcceptPayment, Cashier, PaymentReview и RefundPayment необходимо установить сервис очередей RabbitMQ:

wget https://packages.erlang-solutions.com/erlang-solutions\_1.0\_all.deb

dpkg -i erlang-solutions\_1.0\_all.deb

apt-get update

apt-get install erlang erlang-nox

add-apt-repository 'deb http://www.rabbitmq.com/debian/ testing main'

wget -O- https://www.rabbitmq.com/rabbitmq-release-signing-key.asc | sudo apt-key add –

apt-get update

apt-get install rabbitmq-server

systemctl enable rabbitmq-server

Другие способы установки: <https://www.rabbitmq.com/install-debian.html>

Далее нужно создать пользователя:

abbitmqctl add\_user admin password

sudo rabbitmqctl set\_user\_tags admin administrator

sudo rabbitmqctl set\_permissions -p / admin ".\*" ".\*" ".\*"

Затем запустить сервис:

systemctl enable rabbitmq-server

sudo systemctl start rabbitmq-server

## 3.1.3. Настройка Kubernetes

На начальном этапе необходимо создать «мастер» - виртуальную машину и ноды. Далее необходимо установить ОС Linux Debian 9, sudo, Docker и Kubernetes, используя официальную документацию (<https://kubernetes.io/docs/setup/>).

Далее необходимо произвести настройку, для этого на мастере инициализируем kubeadm панель:

**sudo kubeadm init --pod-network-cidr=10.244.0.0/16.**

cidr выставляется для конкретного плагина, в нашем случае это WeaveNet (взять его можно из официальной документации).

Затем необходимо выполнить:

**mkdir -p $HOME/.kube**

**sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

**sudo chown $(id -u):$(id -g) $HOME/.kube/config**

После этого нужно установить сетевой плагин:

**kubectl apply -f "**[**https://cloud.weave.works/k8s/net?k8s-version=$**](https://cloud.weave.works/k8s/net?k8s-version=$)**(kubectl version | base64 | tr -d '\n')"**

Далее **каждую ноду** необходимо добавить в кластер. Пример команды (Для каждого кластера она разная, в целом kubernetes при выполнении предыдущего пункта пишет ее в конце):

**sudo kubeadm join 172.20.70.210:6443 --token ssot6s.vy0sld2u7kck5v1c --discovery-token-ca-cert-hash sha256:6b2ae90cb485e9f2d89c4b9a62e3e17fe57fc926652e681ce5aa917b68dffa3f**

## 3.1.4. Установка сервисов в кластере Kubernetes

Для работы с сервисами, развернутыми в Kubernetes требуется nginx-сервер со следующими настройками:

server {

listen 80;

listen [: :]:80;

server\_name domain.ru;

charset utf8;

location / {

client\_max\_body\_size 48M;

access\_log /var/log/nginx/domain.ru.log;

error\_log /var/log/nginx/domain.ru-error.log;

proxy\_pass http://my\_proxy;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

proxy\_set\_header Host "domain.ru";

proxy\_read\_timeout 300s;

}

}

upstream my\_proxy {

least\_conn;

server x.x.x.x; # нода1

server x.x.x.x; # нода2

server x.x.x.x; # нодаN

}

Размещение в Kubernetes сервиса **ServiceGate** происходит следующим образом. При помощи CI/CD образ с приложением собирается, помещается в хранилище образов и далее разворачивается в Kubernetes автоматически, путем применения YAML-файлов, расположенных в директории k8s с заменой указанной в них версии приложения на актуальную:

**sed -i 's/{TAG}/'dev'/g' k8s/prod.yaml**

**sed -i 's/{BUILD}/'$CI\_JOB\_ID'/g' k8s/prod.yaml**

**kubectl apply -f k8s/prod.yaml –record**

Репозиторий с проектом: <https://git.zonatelecom.ru/payment-gate-v2/servicegate>

Размещение в Kubernetes нижеуказанных сервисов, происходит аналогично с заменой в файле k8s/prod.yaml значения TAG. Для сервисов:

* PaymentsReview, репозиторий с проектом: <https://git.zonatelecom.ru/payment-gate-v2/paymentsreview>
* CreateOrder, репозиторий с проектом: <https://git.zonatelecom.ru/payment-gate-v2/createorder>
* AcceptPayment, репозиторий с проектом: <https://git.zonatelecom.ru/payment-gate-v2/acceptpayment>
* Cashier, репозиторий с проектом: <https://git.zonatelecom.ru/payment-gate-v2/cashier>
* RefundPayment, репозиторий с проектом: <https://git.zonatelecom.ru/payment-gate-v2/refundpayment>
* Адаптеры платёжных агентов, репозитории:

ЮКасса: <https://git.zonatelecom.ru/payment-gate-v2/payagentsadapters/yookassaadapter>

МКБ: <https://git.zonatelecom.ru/payment-gate-v2/payagentsadapters/mkbadapter>

Payberry: <https://git.zonatelecom.ru/payment-gate-v2/payagentsadapters/payberryadapter>

Сбер.Онлайн: <https://git.zonatelecom.ru/payment-gate-v2/payagentsadapters/sberbankonlineadapter>

Qiwi: <https://git.zonatelecom.ru/payment-gate-v2/payagentsadapters/qiwiadapter>

# 4. ПРОВЕРКА ПРОГРАММЫ

Для проверки работы программы необходимо произвести платеж по инструкции, описанной в документе «Руководство пользователя».

В случае корректного выполнения всех действий, установка программы считается завершённой успешно.

# 5. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

Дополнительные возможности, отличные от приведённых в документах не предусмотрены и могут быть реализованы разработчиком на основании требований эксплуатирующих учреждений.

# 6. ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ОС-операционная система

СУБД-система управления базами данных

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц)  в докум. | №  документа | Входящий № сопрово-дительного документа и дата | Подп. | Дата |
| Изм. | изменен-ных | заменен-ных | новых | аннули-рован-ных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |