Программное обеспечение «New-generation Revenue Management»

ОПИСАНИЕ ФУНКЦИОНАЛЬНЫХ ХАРАКТЕРИСТИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

На 16 листах

2026 год

СОДЕРЖАНИЕ

1. Список сокращений и обозначений

API	*Application programming interface* – интерфейс программирования приложения – описание способов взаимодействия одной компьютерной программы с другими
C/C++	Семейство низкоуровневых языков программирования, обеспечивающих прямой контроль над памятью и максимальную производительность, широко применяемых в системном программировании, встраиваемых системах и высоконагруженных приложениях
Consul	Распределённое, высокодоступное решение для управления сетью сервисов и их конфигурации в динамической распределённой инфраструктуре
CPU	*Central Processing Unit* – «центральное обрабатывающее устройство» – центральный процессор
Golang	Компилируемый многопоточный язык программирования, предназначенный для создания эффективных, масштабируемых и надёжных приложений в сфере серверной разработки, сетевых технологий и облачных сервисов
ОС	Операционная система
Jenkins	Программная система с открытым исходным кодом, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения
Kafka	Распределённая платформа для обработки потоков данных в реальном времени. Она позволяет публиковать, хранить и обрабатывать события, обеспечивая высокую пропускную способность, масштабируемость и отказоустойчивость
Keycloak	Открытая платформа для аутентификации, авторизации и управления доступом к приложениям и сервисам. Она позволяет централизовать управление пользователями, ролями и правами, обеспечивая единый вход (SSO) и упрощая интеграцию с внешними системами

Kubernetes	Открытая платформа для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями. Предназначена для работы с микросервисной архитектурой, распределёнными системами и облачными приложениями, обеспечивает отказоустойчивость, балансировку нагрузки и автоматическое восстановление после сбоев
Nginx	Веб-сервер с открытым исходным кодом, предназначенный для обработки HTTP-запросов, обслуживания статического контента, а также может использоваться как прокси-сервер, балансировщик нагрузки и для других задач
OpenSearch	Распределённая система для поиска, аналитики и визуализации данных с открытым исходным кодом
PostgreSQL	Свободная объектно-реляционная система управления базами данных (СУБД)
Prometheus	Открытая система мониторинга и оповещения с акцентом на сбор, хранение и анализ временных рядов данных. Она предназначена для отслеживания состояния приложений, серверов, инфраструктуры и быстрого реагирования на проблемы
RabbitMQ	Открытый брокер сообщений (message broker) с открытым исходным кодом, предназначенный для асинхронной передачи данных между компонентами приложений. Он выступает посредником, обеспечивая надёжную доставку сообщений, гибкую маршрутизацию и масштабируемость
Redis	*Remote Dictionary Server* – удалённый сервер словарей. Не реляционная система управления базами данных (СУБД) с открытым исходным кодом, которая хранит данные в оперативной памяти в формате «ключ — значение»
RAM	*Random Access Memory* – оперативная память
Runtime-зависимости	Зависимости, которые необходимы для выполнения приложения во время работы, но не требуются для компиляции кода
Zookeeper	программная служба для координации распределённых систем, организованная на основе резидентной базы данных категории «ключ – значение»
ОС	Операционная система
ПО	Программное обеспечение

СУБД	Система управления базами данных – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных

2. Общие сведения

Документ содержит описание функциональных характеристик программного обеспечения программного обеспечения «New-generation Revenue Management» (далее по тексту – ПО).

ПО разработано в виде многофункционального решения, включающего серверное приложение и демонстрационный пользовательский веб-интерфейс, что обеспечивает широкие возможности для пользователей. Веб-приложение обеспечивает доступность из любого места и на любом устройстве, в зависимости от роли пользователя. Жизненный цикл ПО включает все этапы разработки и поддержки, обеспечивая надежную работу и постоянное ее улучшение.

3. Назначение программного обеспечения

ПО предназначено для предоставления операторам связи возможности автоматизации взаиморасчетов с абонентами за услуги связи. ПО обеспечивает комплексное взаимодействие с абонентами. ПО предоставляет операторам API, позволяющий осуществлять следующие функции:

- учет и оценку трафика данных и телефонии;
- прием и обработку платежей абонентов услуг связи;
- выставление счетов за предоставленные услуги связи;
- провидение работы с абонентом;
- создание абонента;
- создание лицевого счета абонента;
- создание продукта с подпиской;
- создание продукта для тарификации событий трафика (телефонии);
- создание продукта с разовым начисления платы (оборудования);
- проведение работы с трафиком;
- осуществление оценки трафика;
- провидение проверки загрузки трафика/просмотра загруженного трафика;
- осуществление работы с лицевым счетом абонента;
- выставление периодического счета;
- предоставление счета по требованию абонента (финального счета);
- отключение продуктов, закрытие лицевого счета абонента.

4. Цели и автоматизируемые функции

4.1. Цели ПО

ПО предназначено для предоставления операторам связи возможности автоматизации взаиморасчетов с абонентами за услуги связи. ПО обеспечивает комплексное взаимодействие с абонентами посредством веб-интерфейса.

4.2. Автоматизируемые функции ПО

ПО позволяет автоматизировать следующие функции:

- учет и оценку трафика данных и телефонии;
- прием и обработку платежей абонентов услуг связи;
- выставление счетов за предоставленные услуги связи.

5. Характеристика функциональной структуры

5.1. Характеристика функциональная структуры

ПО представляет собой серверное приложение без пользовательского интерфейса с API и средство демонстрации экрана в виде веб-приложения, способного формировать запросы к системе и получать ответы. Структура ПО показана на рисунке 1.

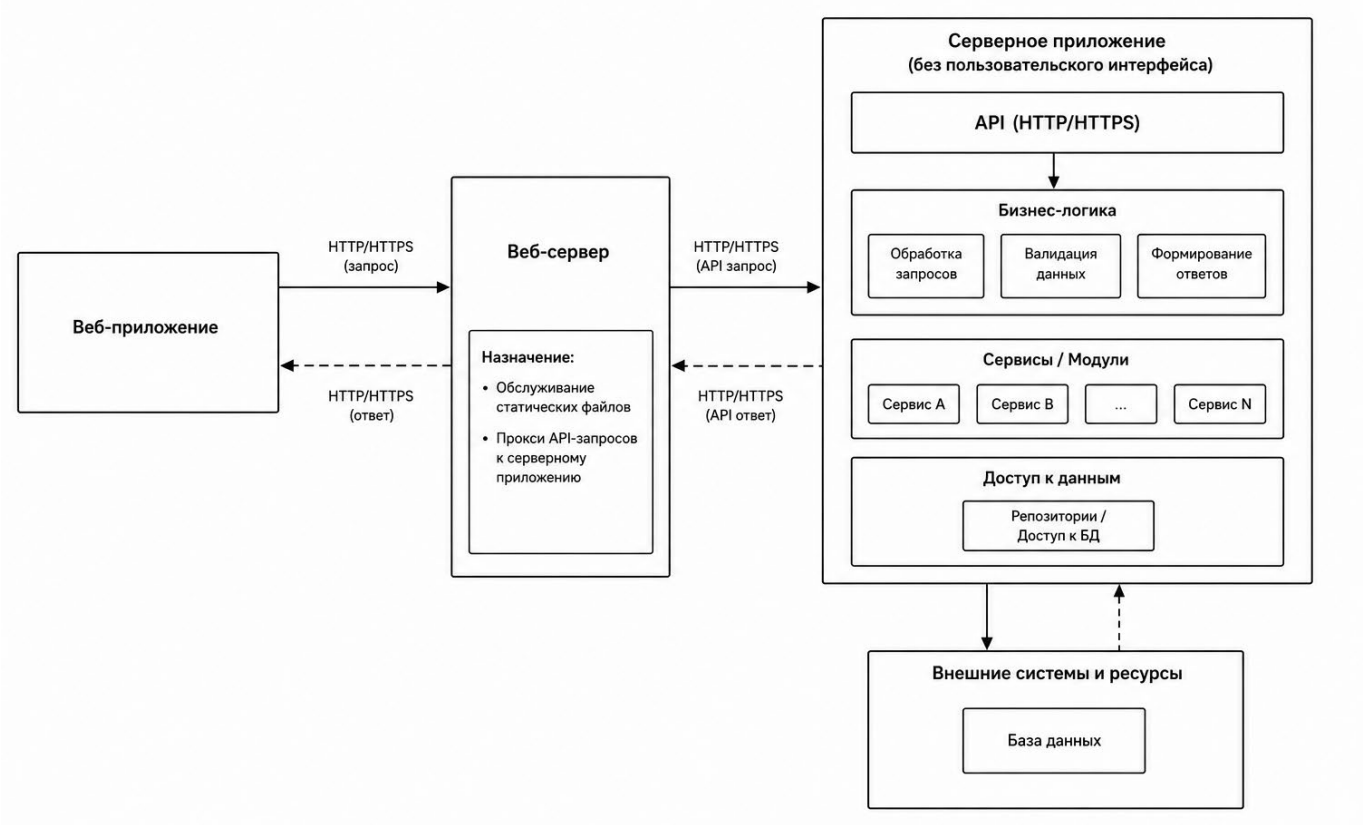

Рисунок 1 – функциональная схема ПО

6. Функциональные задачи ПО

ПО предоставляет операторам связи свой API, позволяющий осуществлять следующие функции:

- провидение работы с абонентом;
- создание абонента;
- создание лицевого счета абонента;
- создание продукта с подпиской;
- создание продукта для тарификации событий трафика (телефонии);
- создание продукта с разовым начисления платы (оборудования);
- проведение работы с трафиком;
- осуществление оценки трафика;
- провидение проверки загрузки трафика/просмотра загруженного трафика;
- осуществление работы с лицевым счетом абонента;
- выставление периодического счета;
- предоставление счета по требованию абонента (финального счета);
- отключение продуктов, закрытие лицевого счета абонента.

7. Архитектура ПО

7.1. Архитектурная структура ПО

В составе ПО существуют следующие уровни:

- серверное приложение

    1) микро-сервисные приложения, написанные на C/C++;

    2) микро-сервисные приложения, написанные на Golang;

- веб-приложение;

- веб-сервер.

7.2. Диаграмма архитектуры

Диаграмма архитектуры представляет взаимодействие между компонентами и уровнями ПО. Диаграмма архитектуры показа на рисунке 2.

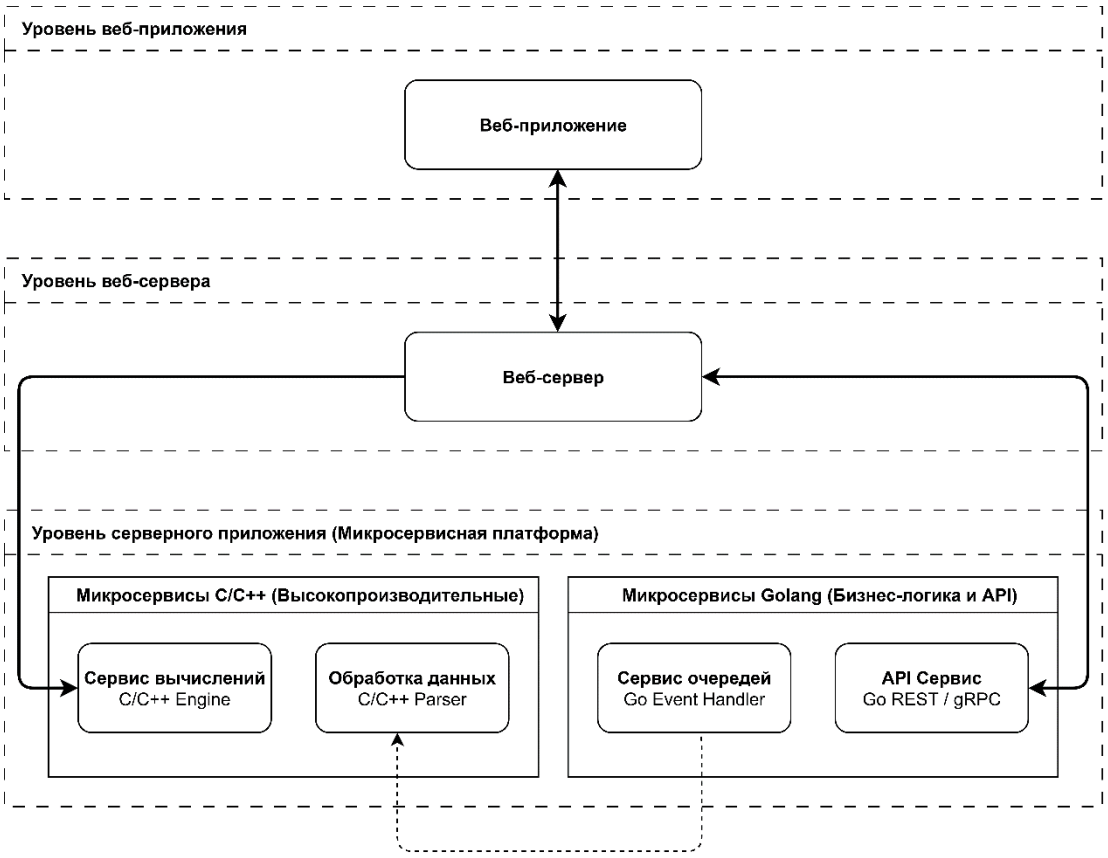

Рисунок 2 – диаграмма архитектуры ПО

8. Аппаратные и программные требования к ПО

ПО предназначено для развертывания в контейнеризированной среде Kubernetes. Kubernetes используется как платформа оркестрации контейнеров, обеспечивающая развертывание, масштабирование и управление контейнеризированными приложениями. Развертывание выполняется с использованием Helm chart/Kubernetes manifests.

8.1. Аппаратные и программные требования

8.1.1. Требования к Kubernetes-кластеру

- Версия Kubernetes: поддерживаемая производителем версия Kubernetes, находящаяся в периоде сопровождения. Рекомендуемая версия: Kubernetes 1.34 и выше либо версия, поддерживаемая выбранной Kubernetes-платформой;
- требования к ОС узлов кластера: Linux-дистрибутив, поддерживаемый выбранной Kubernetes-платформой. Рекомендуемые ОС: Astra Linux;
- требования к container runtime: поддерживается выбранной Kubernetes-платформой. Рекомендуемая: containerd;
- требования к сетевой инфраструктуре: должна быть обеспечена сетевая связность между узлами Kubernetes-кластера, реестром контейнерных образов, доступ к внешним сервисам, входящий доступ пользователей к веб-интерфейсу ПО по протоколу HTTPS, отсутствие блокировок со стороны firewall/ACL/security groups для необходимых сетевых соединений;
- требования к Ingress Controller: должен быть установлен и настроен Ingress Controller, обеспечивающий маршрутизацию входящего HTTP/HTTPS-трафика к сервисам ПО. Рекомендуемый Ingress Controller: nginx-ingress. Ingress Controller должен быть опубликован во внешней или внутренней сети организации через LoadBalancer, NodePort, hostNetwork, reverse proxy или иной утвержденный способ публикации. Для доступа пользователей должно быть настроено DNS-имя, указывающее на адрес Ingress Controller;

- требования к StorageClass/PersistentVolume: в Kubernetes-кластере должен быть настроен StorageClass, обеспечивающий динамическое выделение PersistentVolume для PersistentVolumeClaim компонентов ПО. Также возможно использовать заранее созданные PersistentVolume при условии соответствия требованиям PVC приложения по размеру, access mode и storageClassName;

- требования к реестр контейнерных образов: поддержка работы с Bearer token-схемой авторизации и с OCI форматом образов. Рекомендуемые: Sonatype Nexus Repository.

8.1.2. Аппаратные требования к узлам кластера

Минимальная конфигурация кластера:

- количество master-узлов: 3 шт.;
- количество worker-узлов: 3 шт.;
- количество frontend-узлов: 2 щт.;
- количество узлов мониторинга: 1 шт.

Минимальные требования к ресурсам узлов

- master-, worker-, мониторинг-узел:
    1) CPU (шт.): 4;
    2) RAM (ГБ): 8;
    3) объем диска (ГБ): 60.
- frontend-узел:
    1) CPU (шт.): 2;
    2) RAM (ГБ): 4;
    3) объем диска (ГБ): 50.

Для всех узлов следует выделять быстрые диски (400+ IOPS).

8.1.3. Ресурсные требования компонентов ПО

Компоненты ПО разворачиваются в Kubernetes в виде Pod'ов. Для каждого компонента должны быть заданы requests и limits по CPU и RAM. При использовании

постоянного хранения данных должны быть заданы PersistentVolumeClaim требуемого размера.

Минимальные значения requests/limits и размеры PVC определяются составом развертываемых компонентов и профилем нагрузки.

8.1.4. Runtime-зависимости

Для функционирования ПО могут использоваться следующие программные компоненты:

- система аутентификации и авторизации: Keycloak 18.0.0 и выше;
- СУБД: PostgreSQL 14 и выше;
- хранилище данных / поисковый движок: OpenSearch 1.3.6 и выше;
- брокер сообщений: Kafka + Zookeeper и/или RabbitMQ;
- key-value хранилище / service discovery: Consul;
- cache: Redis;
- объектное хранилище: S3-compatible storage.

Фактический набор обязательных зависимостей определяется конфигурацией ПО и используемыми функциональными модулями.

Эксплуатационные компоненты

- система мониторинга: Prometheus 2.42.0 и выше;
- система CI/CD: Jenkins.

Prometheus используется для мониторинга состояния ПО и сбора метрик. Jenkins используется для автоматизации сборки и развертывания ПО и не является обязательным компонентом runtime-среды, если ПО после установки функционирует без Jenkins.

9. Численность, функции и квалификация персонала, работающего в ПО

Численность персонала определяются в рамках проекта внедрения.

Функции персонала, работающего в ПО:

- Проведение биллинговых операций;
- операционная поддержка биллинговых операций, включая применение датафиксов;
- установка обновлений ПО;
- администрирование ПО и элементов инфраструктуры, на которых оно развернуто.

10. Режим функционирования программного обеспечения

В основном режиме функционирования ПО обеспечивает:

- беспрерывную работу в режиме – 24 часа в сутки, 7 дней в неделю (24х7);
- выполнение всех функций в полном объеме; за исключением периодов проведения профилактических и других работ, а также устранения возникших нештатных ситуаций.

В случае возникновения внештатных ситуаций организована возможность восстановления работоспособности ПО путем отката до последней рабочей версии с сохранением целостности информации.