Описание функциональных характеристик

Программа «NORA — многопротокольный реестр артефактов»

Правообладатель: ООО «ТАИАРС» · Автор: Волков Павел Сергеевич · Версия программы: 1.0.0 · Дата: 2026-06-05

1. Назначение программы

Программа «NORA» предназначена для централизованного хранения, кэширования и распространения программных компонентов (артефактов), которые используются при разработке, сборке и развёртывании программного обеспечения. Программа решает следующие задачи:

- Организация внутреннего репозитория программных зависимостей.
- Проксирование и кэширование общедоступных репозиториев с сохранением полученных артефактов в локальное хранилище — для ускорения сборки и обеспечения доступности при отсутствии подключения к сети Интернет.
- Контроль целостности артефактов и защита цепочки поставок программного обеспечения.
- Курирование зависимостей: блокировка, разрешение и изоляция пространств имён.
- Аудит и протоколирование всех операций доступа к артефактам.

2. Функциональные возможности

2.1. Многопротокольная поддержка

Программа обеспечивает работу с артефактами по 16 протоколам:

Протокол	Стандарт	Назначение
Docker / OCI	OCI Distribution Specification v1.0	Контейнерные образы
Helm OCI	OCI Distribution Specification v1.0	Helm-чарты (Kubernetes)
npm	npm Registry API	Библиотеки JavaScript и TypeScript
Maven	Maven Repository Layout	Библиотеки Java и Kotlin
PyPI	PEP 503 (Simple Repository API)	Библиотеки Python
Cargo	Cargo Alternative Registry Protocol	Библиотеки Rust
Go	Go Module Proxy Protocol (GOPROXY)	Модули Go
NuGet	NuGet V3 API	Библиотеки .NET
RubyGems	RubyGems API	Библиотеки Ruby
Terraform	Terraform Provider Registry Protocol	Модули и провайдеры Terraform
Ansible Galaxy	Ansible Galaxy API	Коллекции и роли Ansible
Pub	Pub Repository Spec	Библиотеки Dart и Flutter
Conan	Conan v2 API	Библиотеки C и C++

Протокол	Стандарт	Назначение
APT	Debian Repository Format	Пакеты deb (Debian, Ubuntu)
YUM	RPM Repository Format (repomd)	Пакеты rpm (RHEL, CentOS, ALT Linux)
Raw	HTTP PUT/GET	Произвольные файлы

2.2. Режимы работы

- **Хранилище (hosted).** Приём и хранение артефактов, опубликованных пользователями (docker push, npm publish, mvn deploy, PUT /raw/). Опубликованные артефакты записываются в локальное хранилище на файловой системе сервера.
- **Прокси-кэш (proxy).** Прозрачное проксирование запросов к внешним репозиториям. При первом обращении артефакт загружается из внешнего источника и **сохраняется (кэшируется) в локальное хранилище на файловой системе сервера**; последующие обращения обслуживаются из локальной файловой системы без повторного обращения к внешнему источнику.
- **Комбинированный.** Одновременная работа в режимах хранилища и прокси-кэша: сначала проверяется наличие артефакта в локальном хранилище, затем выполняется обращение к внешнему репозиторию.
- **Виртуальный (group).** Объединение нескольких репозиториев (hosted и proxy) в единую точку доступа с одним URL.

2.3. Размещение и хранение данных

Все артефакты (как опубликованные пользователями, так и полученные при проксировании), метаданные и журналы аудита сохраняются в локальное хранилище **на файловой системе сервера**. Структура хранения иерархическая: `{протокол}/{пакет}/{артефакт}`. В качестве альтернативы поддерживается S3-совместимое объектное хранилище (MinIO, Yandex Object Storage, Selectel S3). Все данные размещаются на технических средствах, находящихся на территории Российской Федерации.

2.4. Предварительное кэширование (nora mirror)

Команда `nora mirror` обеспечивает предварительную загрузку зависимостей проекта через прокси-кэш программы с сохранением в локальное хранилище. Поддерживаются файлы фиксации версий: `package-lock.json` (npm, версии 1–3), `requirements.txt` (pip/Python), `Cargo.lock` (Cargo/Rust), вывод `mvn dependency:list` (Maven/Java). Поддерживается указание списка пакетов с загрузкой всех версий (`--all-versions`) и параллельная загрузка.

2.5. Управление доступом

Ролевая модель: `read` (чтение, по умолчанию), `write` (чтение и публикация), `admin` (полный доступ, включая управление пользователями). Механизмы аутентификации:

- Файл htpasswd с хешированием паролей алгоритмом Argon2id с зануливанием памяти после использования.
- Единый вход через OpenID Connect (OIDC) — Keycloak, Azure AD, Google и др.

Для доступа через API используются токены с ограниченным сроком действия.

2.6. Контроль целостности

При проксировании программа вычисляет и сохраняет контрольную сумму SHA-256 для каждого артефакта; при повторной выдаче из кэша сумма проверяется, при расхождении запрос отклоняется. При публикации выполняется валидация имён файлов (защита от обхода каталогов), проверка соответствия имени пакета в URL и теле запроса, контроль иммутабельности версий.

2.7. Курирование зависимостей

- Blocklist — блокировка конкретных пакетов или пространств имён.
- Allowlist — разрешение только проверенных зависимостей.
- Namespace isolation — изоляция областей видимости между командами.
- Min-release-age — карантин новых пакетов (по умолчанию 24 часа) для защиты от атак на цепочку поставок.

Режим fail-closed: при ошибке проверки пакет блокируется.

2.8. Автоматический выключатель и аудит

- **Circuit breaker.** При недоступности внешних реестров после заданного числа сбоев программа прекращает обращения к недоступному реестру и обслуживает запросы из кэша.
- **Аудит.** Все операции фиксируются в файле аудита формата JSON Lines (временная метка, тип операции, протокол, идентификатор запроса).

2.9. Политики жизненного цикла

- Retention policies — автоматическое удаление по возрасту (max_age) и числу версий (max_versions).
- Cleanup policies — периодическая очистка по расписанию с режимом предварительного просмотра.

2.10. Мониторинг и эксплуатация

Эндпоинты `/health` (работоспособность и доступность хранилища), `/ready` (готовность), `/metrics` (метрики Prometheus); готовые дашборды Grafana. Эксплуатационные команды: резервное копирование (`nora backup` / `nora restore`), сборка мусора (`nora gc`), миграция хранилища (`nora migrate --from local --to s3`), миграция из внешних систем (`nora migrate --from nexus|artifactory|gitlab`), горячая перезагрузка конфигурации, потоковая загрузка артефактов.

2.11. Веб-интерфейс

Встроенный веб-интерфейс для просмотра содержимого реестра (список артефактов по протоколам, количество версий и размер, журнал операций, статистика загрузок) на русском и английском языках. Доступна интерактивная документация API в формате OpenAPI 3.0.

3. Конфигурация

Конфигурация выполняется через переменные окружения (префикс `NORA_`), файл `config.toml` или их комбинацию. Приоритет: переменные окружения, файл конфигурации, значения по умолчанию. Поддерживается горячая перезагрузка конфигурации без перезапуска сервиса.

4. Технические характеристики

Язык реализации	Rust
Объём исходного кода	13 084 строки
Формат поставки	Статически слинкованный исполняемый файл, Docker-образ, Helm-чарт, deb- и rpm-пакеты
Размер бинарного файла	< 23 МБ
Поддерживаемые ОС	Astra Linux, РЕД ОС, ALT Linux, Debian, Red Hat Enterprise Linux, Ubuntu (ядро Linux 4.15 и выше)
Архитектуры	x86_64, aarch64
Контейнеризация	Docker-образ (базовый образ scratch)
Системная интеграция	systemd (файл сервиса в комплекте)
Хранилище	Локальная файловая система или S3-совместимое объектное хранилище
Потребление памяти	< 100 МБ при типичной нагрузке
Время запуска	< 3 с
Лицензия	MIT

5. Перечень компонентов

Программа использует 238 библиотек с открытым исходным кодом. Полный перечень компонентов с указанием наименования, версии и лицензии представлен в файле SBOM (формат CycloneDX — `nora.cdx.json`, а также формат SPDX — `nora.spdx.json`). Все используемые библиотеки проверены инструментом cargo audit — уязвимости не обнаружены.