# Платформа автоматизации с ИИ-агентами «Nodul»

## Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения

---

### АННОТАЦИЯ

Документ содержит описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения «Платформа автоматизации с ИИ-агентами Nodul» (далее — «ПО», «Nodul»), в том числе процессы устранения неисправностей, выявленных в ходе эксплуатации, процессы совершенствования ПО, а также информацию о персонале, необходимом для обеспечения такой поддержки.

Документ предназначен для пользователей программного обеспечения, сотрудников организации-разработчика, а также лиц, оценивающих ПО на соответствие требованиям к программному обеспечению, включаемому в Единый реестр российских программ для электронных вычислительных машин и баз данных.

---

### СОДЕРЖАНИЕ



---

# 1. ОБЩИЕ СВЕДЕНИЯ

## 1.1. Наименование программы

Полное наименование программы: «Платформа автоматизации с ИИ-агентами Nodul».

В рамках настоящего документа употребляются также обозначения «Nodul» и «ПО».

## 1.2. Назначение документа

Настоящий документ описывает совокупность процессов, организационных мер и технических средств, применяемых разработчиком ПО для:

- поддержания работоспособности ПО на протяжении всего срока его эксплуатации;
- устранения неисправностей, выявленных в ходе эксплуатации ПО;
- совершенствования ПО — выпуска новых версий с расширенной функциональностью и улучшениями;
- информационного и технического взаимодействия с пользователями ПО.

Документ носит описательный характер и применим как к облачной эксплуатации ПО, так и к эксплуатации в варианте поставки on-premise — в инфраструктуре заказчика.

---

# 2. ПРОЦЕССЫ, ОБЕСПЕЧИВАЮЩИЕ ПОДДЕРЖАНИЕ ЖИЗНЕННОГО ЦИКЛА ПО

## 2.1. Общая модель жизненного цикла

Поддержание жизненного цикла ПО обеспечивается силами разработчика и включает следующие основные процессы:

- проектирование, разработка и тестирование новых версий ПО;
- выпуск версий и поставка ПО конечным пользователям;
- сопровождение и техническая поддержка пользователей;
- регистрация, диагностика и устранение неисправностей, выявленных в ходе эксплуатации;
- совершенствование ПО на основе обратной связи от пользователей и собственного бэклога развития;
- обеспечение информационной безопасности на всех этапах жизненного цикла.

Процессы применяются непрерывно: новые версии ПО выпускаются по мере накопления изменений; обращения пользователей принимаются в режиме, описанном в разделе 2.4; устранение неисправностей выполняется в порядке приоритета и интегрируется в текущий цикл выпуска версий.

## 2.2. Процессы разработки и выпуска новых версий

Разработка ПО ведётся по итеративной модели с короткими циклами выпуска. Основные элементы процесса:

- управление требованиями: входящие пожелания, обращения и предложения пользователей фиксируются в системе управления задачами, приоритизируются и включаются в план развития;
- разработка и ревью кода: изменения вносятся в единую систему контроля версий, проходят обязательное код-ревью силами других участников команды разработки;

- автоматизированное тестирование: для критичных компонентов и сценариев используются модульные и интеграционные тесты, выполняемые в системе непрерывной интеграции при каждом изменении;
- ручное тестирование функциональности на тестовых окружениях, в том числе проверка узлов сценариев, интеграций, ИИ-агентов и инструментов отладки;
- сборка релизных артефактов: исходный код собирается в контейнерные образы, проходит проверку и помечается версионным тегом;
- выпуск версии: подготовленные артефакты публикуются в репозиторий контейнерных образов; для облачной эксплуатации обновление выполняется централизованно средствами оркестратора Kubernetes, для эксплуатации on-premise обновлённые Helm-чарты предоставляются эксплуатирующей стороне.

Изменения, не затрагивающие функциональность и совместимость (исправления ошибок, оптимизации), могут выпускаться чаще, чем функциональные обновления, и применяются автоматически в облачной инсталляции.

## 2.3. Процессы поставки и развёртывания

ПО поставляется в следующих вариантах:

- **облачная эксплуатация** — пользователь получает доступ к ПО через веб-браузер по сети Интернет; развёртывание и обновление обеспечиваются разработчиком; от пользователя требуется только регистрация рабочего пространства и подключение собственных учётных записей внешних сервисов;
- **on-premise — в инфраструктуре заказчика** — серверная часть разворачивается средствами Kubernetes; поставка осуществляется в виде Helm-чартов и набора контейнерных образов; вместе с дистрибутивом передаётся эксплуатационная документация, описывающая установку, начальную конфигурацию и регламентное обслуживание.

В обоих вариантах поставки клиентская часть исполняется в браузере пользователя и не требует установки дополнительного ПО на рабочее место.

Для on-premise-инсталляций обновлённые версии чартов и контейнерных образов предоставляются заказчику по согласованному каналу; обновление в инфраструктуре заказчика выполняется силами эксплуатационной службы заказчика средствами Helm и Kubernetes.

## 2.4. Процессы сопровождения и технической поддержки пользователей

Сопровождение пользователей ПО обеспечивается силами разработчика и включает:

- приём и регистрацию обращений пользователей;
- классификацию обращений по типу (вопрос по эксплуатации, инцидент, предложение по развитию) и по приоритету (критичные инциденты, обычные обращения);
- диагностику обращения: ответ пользователю с разъяснением, запрос дополнительной информации либо передача обращения в работу команде разработки;
- информирование пользователя о статусе обращения и предоставление решения или обходного пути;
- закрытие обращения по подтверждению пользователя.

### Каналы поддержки

Основные каналы взаимодействия пользователей со службой поддержки разработчика:

- форум сообщества Nodul: https://community.nodul.ru — основной канал для регистрации обращений и обсуждения сценариев использования;

- Telegram-сообщество: https://t.me/nodul — для оперативных вопросов и взаимодействия с сообществом и разработчиком;
- электронная почта: help@nodul.ru — для случаев недоступности аккаунта, критических инцидентов и обращений, требующих приватного канала.

**Режим работы службы технической поддержки**

Режим работы службы технической поддержки разработчика: с понедельника по пятницу с 10:00 до 22:00 (центральноевропейское время; по московскому времени — с 11:00 до 23:00). В выходные и нерабочие праздничные дни обращения обрабатываются по возможности; критические инциденты, поступающие через электронную почту, рассматриваются вне зависимости от рабочего расписания.

**Документация и самостоятельная диагностика**

Помимо обращения в службу поддержки, пользователю предоставлены средства для самостоятельной диагностики и обучения:

- эксплуатационная документация продукта, опубликованная по адресу https://documentation.nodul.ru;
- раздел «Возможные ошибки» в документации, описывающий типовые ошибки сборки сценариев, ошибки интеграций и ошибки платформы с рекомендациями по устранению;
- встроенный ИИ-ассистент сборки и отладки сценариев (AI scenario helper), способный объяснять, проектировать и отлаживать сценарии в режиме диалога;
- журнал истории выполнения с входными и выходными данными, длительностями и сообщениями об ошибках по каждому узлу сценария, позволяющий пользователю самостоятельно локализовать причину сбоя в сценарии.

## 2.5. Процессы устранения неисправностей, выявленных в ходе эксплуатации

Под неисправностью понимается отклонение фактического поведения ПО от ожидаемого, описанного в эксплуатационной документации, либо отказ компонента ПО.

Процесс устранения неисправностей включает:

1. **Регистрация.** Сообщение о неисправности поступает от пользователя по одному из каналов поддержки (см. п. 2.4) либо выявляется внутренними средствами мониторинга и наблюдаемости. Каждое сообщение фиксируется в системе управления задачами.
2. **Классификация и приоритизация.** Сообщение классифицируется как ошибка, инцидент эксплуатации или запрос на изменение. Назначается приоритет, учитывающий влияние на работу пользователей, наличие обходного пути и количество затронутых пользователей.
3. **Диагностика.** Инженер технической поддержки или разработчик выполняет воспроизведение неисправности, анализ журналов исполнения сценариев и системных журналов, при необходимости запрашивает у пользователя дополнительные сведения.
4. **Устранение.** Для критичных инцидентов в первую очередь предоставляется обходное решение (workaround) и/или производится оперативная корректировка на стороне облачной инсталляции. Постоянное исправление ошибки вносится в исходный код, проходит кодревью и автоматизированное тестирование, после чего включается в очередной выпуск версии ПО.
5. **Доставка исправления.** Для облачной эксплуатации исправление становится доступным пользователю автоматически после очередного обновления компонентов. Для эксплуата-

ции on-premise исправление поставляется в составе обновлённого Helm-чарта и контейнерных образов; применение обновления выполняется эксплуатационной службой заказчика.

6. **Информирование и закрытие.** После устранения неисправности пользователь информируется о результате; обращение закрывается.

Конкретные сроки реакции на обращения и сроки устранения неисправностей не нормируются настоящим документом и могут быть зафиксированы отдельным соглашением (договором, SLA) с конкретным заказчиком.

## 2.6. Процессы совершенствования и развития ПО

Совершенствование ПО осуществляется на основе:

- обратной связи пользователей (обращения в каналах поддержки, обсуждения на форуме сообщества, прямые предложения);
- внутреннего бэклога развития продукта, который ведётся командой разработки и учитывает рыночные требования, появление новых технологий (например, новых моделей LLM, новых протоколов взаимодействия с ИИ-инструментами) и архитектурные улучшения;
- результатов анализа эксплуатации (типичные ошибки сценариев, узкие места производительности, наиболее востребованные интеграции).

Совершенствование выполняется в виде:

- расширения функциональности (новые узлы, новые интеграции, новые возможности ИИ-агентов, поддержка новых моделей и провайдеров);
- улучшений удобства использования (доработки интерфейса, ИИ-ассистента сборки сценариев, средств отладки);
- оптимизации производительности и масштабируемости компонентов;
- актуализации используемых библиотек, рантаймов и базовых образов.

Изменения попадают в плановый цикл разработки (см. п. 2.2). Информация о значимых изменениях публикуется в эксплуатационной документации и анонсируется в каналах сообщества.

## 2.7. Процессы обеспечения информационной безопасности при сопровождении

При выполнении процессов жизненного цикла соблюдаются следующие принципы:

- разграничение доступа сотрудников разработчика к производственным средам и пользовательским данным — только в объёме, необходимом для выполнения трудовых обязанностей;
- доступ сотрудников к коду, инфраструктуре и системам управления задачами осуществляется с использованием персональных учётных записей;
- учётные данные внешних сервисов, передаваемые пользователями платформе (авторизации, токены, API-ключи), хранятся отдельно от тела сценариев и доступны только в момент исполнения;
- зависимости и базовые контейнерные образы регулярно обновляются с учётом информации об уязвимостях; критичные обновления безопасности приоритизируются и выпускаются вне общего цикла;
- при возникновении инцидента информационной безопасности затронутые пользователи уведомляются по каналам поддержки.

# 3. ИНФРАСТРУКТУРА, ИСПОЛЬЗУЕМАЯ ДЛЯ ПОДДЕРЖАНИЯ ЖИЗНЕННОГО ЦИКЛА ПО

## 3.1. Среды разработки, тестирования и эксплуатации

Для поддержания жизненного цикла используются логически разделённые среды:

- **среда разработки** — для написания, проверки и локального запуска кода;
- **среда тестирования (preview/staging)** — для проверки собранных версий ПО на стендах, максимально приближённых к производственной конфигурации;
- **среда эксплуатации (production)** — облачная инсталляция ПО, доступная пользователям по сети Интернет, либо инфраструктура заказчика в случае on-premise-эксплуатации.

Изменения проходят через все среды последовательно: попадание изменения в среду эксплуатации возможно только после успешного прохождения автоматизированных и ручных проверок в предыдущих средах.

## 3.2. Системы управления исходным кодом и сборки

- единая система контроля версий с обязательным код-ревью и историей изменений;
- система непрерывной интеграции и доставки (CI/CD), выполняющая сборку контейнерных образов, автоматизированные проверки и публикацию артефактов;
- репозиторий контейнерных образов, используемый как для облачной эксплуатации, так и для поставки on-premise.

## 3.3. Системы регистрации обращений и управления задачами

- каналы приёма обращений пользователей (см. п. 2.4) интегрированы с внутренней системой управления задачами разработчика; каждое обращение, требующее доработки или исправления, фиксируется в виде задачи и связывается с релизной поставкой;
- ведётся внутренний бэклог развития продукта; приоритеты пересматриваются командой разработки регулярно.

## 3.4. Хранение и распространение дистрибутивов

- исходные коды и сборочные конфигурации хранятся в системе контроля версий;
- бинарные артефакты (контейнерные образы) хранятся в репозитории контейнерных образов;
- Helm-чарты для on-premise-поставки версионируются и предоставляются заказчику по согласованному каналу вместе с эксплуатационной документацией;
- эксплуатационная документация ПО публикуется по адресу https://documentation.nodul.ru и поддерживается в актуальном состоянии.

---

# 4. ИНФОРМАЦИЯ О ПЕРСОНАЛЕ, НЕОБХОДИМОМ ДЛЯ ОБЕСПЕЧЕНИЯ ПОДДЕРЖКИ ПО

## 4.1. Состав ролей

Для поддержания жизненного цикла ПО разработчик располагает квалифицированной командой, включающей следующие функциональные роли:

- **разработчики серверной части** — проектирование и сопровождение API, движка исполнения сценариев, сервисов интеграций, подсистемы хранения;

- **разработчики клиентской части** — реализация и сопровождение веб-интерфейса, визуального конструктора сценариев, отладчика, встроенного ИИ-ассистента;
- **инженеры по интеграциям** — разработка и сопровождение готовых узлов для внешних сервисов, узлов работы с ИИ-моделями, поддержка протокола MCP, доработка узлов под изменения внешних API;
- **инженеры по эксплуатации (DevOps / SRE)** — сопровождение Kubernetes-кластеров, конвейеров сборки и доставки, систем наблюдаемости, обеспечение отказоустойчивости и масштабирования;
- **инженеры по обеспечению качества** — формирование и поддержка автоматизированных и ручных проверок, регрессионное тестирование релизных кандидатов;
- **специалисты технической поддержки** — приём, регистрация и обработка обращений пользователей, взаимодействие с разработкой по инцидентам и запросам;
- **технические писатели и контент-менеджеры** — подготовка и актуализация эксплуатационной документации, материалов сообщества;
- **продуктовые менеджеры и руководители разработки** — приоритизация задач, координация выпуска версий, развитие продукта.

Численность сотрудников в каждой роли определяется текущими потребностями развития и сопровождения ПО и обеспечивает выполнение всех процессов, описанных в разделе 2.

### 4.2. Требования к квалификации

К сотрудникам, участвующим в поддержании жизненного цикла ПО, предъявляются следующие общие требования:

- профильное образование (высшее или средне-специальное в области информационных технологий) либо подтверждённый практический опыт работы по соответствующей роли;
- владение применяемыми технологиями и инструментами: для разработчиков — TypeScript / JavaScript, Node.js, современные веб-фреймворки, СУБД (MongoDB, MySQL); для инженеров по эксплуатации — Kubernetes, Helm, средства наблюдаемости и систем непрерывной интеграции; для инженеров технической поддержки — общее понимание архитектуры веб-приложений и принципов работы интеграций и ИИ-агентов;
- понимание принципов безопасной разработки и работы с конфиденциальными данными пользователей;
- готовность работать в режиме итеративной разработки и оперативно реагировать на пользовательские обращения.

### 4.3. Поддержание компетенций

Поддержание компетенций сотрудников обеспечивается:

- внутренней передачей знаний при адаптации новых сотрудников и обмене опытом внутри команд;
- регулярной работой с актуальными версиями используемых технологий и базовых компонентов (рантаймов, СУБД, оркестратора Kubernetes);
- участием сотрудников в развитии собственного продукта и сопровождении его в эксплуатации, что обеспечивает непосредственное практическое освоение применяемых решений;
- актуализацией внутренней технической документации (архитектурных описаний, регламентов поддержки, инструкций по эксплуатации компонентов).

---

## 5. КАНАЛЫ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЯМИ И КОНТАКТНАЯ ИНФОРМАЦИЯ

Для взаимодействия с пользователями по вопросам сопровождения, развития и эксплуатации ПО разработчиком предоставляются следующие каналы:

| Канал | Адрес | Назначение |
|---|---|---|
| Форум сообщества | https://community.nodul.ru | Основной канал регистрации обращений, обсуждения сценариев использования и обмена опытом между пользователями |
| Telegram-сообщество | https://t.me/nodul | Оперативные вопросы, новости продукта, взаимодействие с сообществом и разработчиком |
| Электронная почта | help@nodul.ru | Приватный канал: недоступность аккаунта, критические инциденты, обращения, требующие конфиденциальности |
| Эксплуатационная документация | https://documentation.nodul.ru | Справочные материалы по работе с ПО, описание узлов, интеграций, типовых ошибок и их устранения |

Режим работы службы технической поддержки разработчика: с понедельника по пятницу с 10:00 до 22:00 (центральноевропейское время; по московскому времени — с 11:00 до 23:00). В выходные и нерабочие праздничные дни обращения обрабатываются по возможности; критические инциденты, поступающие через электронную почту, рассматриваются вне зависимости от рабочего расписания.