06_описание_жизненного_цикла

Описание жизненного цикла ПО “Аралтыш”

Общая модель жизненного цикла

Жизненный цикл ПО включает:

1. Планирование изменений.
2. Разработку.
3. Проверку.
4. Выпуск версии.
5. Публикацию артефактов.
6. Развертывание.
7. Эксплуатацию.
8. Сопровождение.
9. Вывод версии из эксплуатации.

Разработка

Исходный код хранится в системе контроля версий Git.

Рабочая ветка: `main`.

Изменения фиксируются коммитами с описанием назначения изменения.

Перед выпуском версии выполняется:

- проверка синтаксиса Python;
- проверка отсутствия секретов в репозитории;
- сборка контейнерного образа;
- проверка `/health`;
- ручная проверка ключевых сценариев.

Версионирование

Используется семантическое версионирование `MAJOR.MINOR.PATCH`.

Пример: `0.1.7`.

Каждый релиз помечается Git-тегом `vX.Y.Z`.

Сборка

Основной способ сборки:

```
docker build -t araltysh:X.Y.Z .
```

Контейнерный образ содержит:

- Python runtime;
- зависимости из `requirements.txt`;
- `app.py`;
- статические материалы из `assets/`.

Выпуск версии

Процедура выпуска:

1. Обновить версию в `package.json`.
2. Обновить документацию при необходимости.
3. Выполнить локальную проверку.
4. Создать Git commit.
5. Создать Git tag.
6. Собрать контейнерный образ.
7. Опубликовать артефакты.
8. Создать release notes.

Сопровождение

Сопровождение включает:

- исправление ошибок;
- актуализацию зависимостей;
- обновление документации;
- обработку обращений пользователей;
- проверку совместимости с MAX Bot API;
- контроль безопасности секретов и конфигурации.

Совершенствование и модернизация

Совершенствование ПО выполняется правообладателем в рамках плановых и внеплановых изменений.

Основаниями для модернизации являются:

- обращения пользователей и операторов экземпляров сервиса;
- выявленные ошибки и инциденты эксплуатации;
- изменения MAX Bot API;
- требования безопасности;
- необходимость повышения надежности, производительности и удобства пользовательских сценариев;
- изменение нормативных, организационных или эксплуатационных требований.

Процесс совершенствования включает:

1. Фиксацию задачи или предложения в рабочем списке изменений.
2. Анализ влияния изменения на сценарии родителя, ребенка и администратора.
3. Разработку изменения в исходном коде и документации.
4. Проверку синтаксиса и базовых сценариев.
5. Проверку миграций и совместимости с текущей базой данных, если изменение затрагивает данные.
6. Сборку контейнерного образа.
7. Выпуск новой версии с Git-тегом `vX.Y.Z`.
8. Публикацию артефактов и release notes.
9. Обновление эксплуатационной документации при изменении порядка установки, настройки или использования.

Плановые изменения группируются в релизы. Срочные исправления безопасности или критичных ошибок могут выпускаться вне планового цикла отдельной patch-версией.

Устранение неисправностей

Неисправность фиксируется по обращению пользователя, сообщению администратора, результатам мониторинга или ручной проверке. Для каждой неисправности определяется влияние на эксплуатацию:

- критичная: сервис недоступен или не отправляет тревожные сообщения;
- значимая: нарушен отдельный сценарий пользователя или администратора;
- обычная: ошибка не блокирует основные функции.

Порядок устранения:

1. Проверить доступность `/health`.
2. Проверить журналы контейнера.
3. Проверить настройки `MAX_BOT_TOKEN`, `MAX_WEBHOOK_URL` и доступность MAX Bot API.
4. Проверить подключение к PostgreSQL.
5. Воспроизвести сценарий на тестовом экземпляре.

6. Подготовить исправление, проверить его и выпустить обновленную версию.
7. Обновить эксплуатационный экземпляр после резервного копирования.

Информация о значимых исправлениях отражается в release notes и, при необходимости, в пользовательской или административной документации.

Персонал сопровождения

Минимальный состав персонала для сопровождения базового экземпляра:

- разработчик/сопровождающий - анализ ошибок, внесение изменений, выпуск версий;
- администратор экземпляра - настройка окружения, домена, базы данных, резервного копирования и обновлений;
- специалист поддержки - прием обращений пользователей и передача технических вопросов разработчику.

В малом внедрении эти роли может выполнять один специалист или правообладатель. Для регионального или организационного внедрения роли могут быть распределены между сотрудниками оператора экземпляра.

Хранение артефактов

Для заявки в реестр рекомендуется хранить:

- исходный код;
- архив исходного кода релиза;
- контейнерный образ;
- checksums артефактов;
- release notes;
- протоколы испытаний.

Основной контур хранения для реестра должен быть размещен на территории Российской Федерации. Публичные зарубежные зеркала могут использоваться как дополнительные каналы распространения.

Вывод версии из эксплуатации

Версия выводится из эксплуатации после:

- выпуска более новой стабильной версии;
- переноса пользователей на новую версию;
- проверки работоспособности новой версии;
- сохранения архива старой версии.