АКС Комплекс

Описание жизненного цикла

Программный комплекс автоматизации и контроля процесса подбора и учета продукции «АКС Комплекс»

Версия: 1.0
Дата: 27.10.2026
Правообладатель: ИП Краснов Владимир Андреевич
ИНН: 781422796858 · **ОГРНИП:** 320784700096770
Свидетельство: № 2025687159

1. Общие положения

Настоящий документ описывает жизненный цикл программного комплекса автоматизации и контроля процесса подбора и учета продукции «АКС Комплекс» (далее — Комплекс): стадии создания, выпуска, внедрения, эксплуатации, сопровождения, модернизации и вывода из эксплуатации.

Документ распространяется на все компоненты Комплекса:

Компонент	Версия	Назначение
`ascs_api`	2.0	Ядро: заказы, маршруты, очередь, API, печать, статистика
`ascs_ui`	1.0	АРМ диспетчера
`ptl_api`	3.0	Управление PTL-модулями
`rid_api`	3.0	Управление информационными дисплеями
`lighthouse_api`	3.0	Управление световыми колоннами

Версия Комплекса в целом — **1.0**.

2. Стадии жизненного цикла

2.1. Анализ требований

- Сбор требований заказчика и анализ складских процессов.
- Формализация сценариев сортировки, наборки, опта и отгрузки.
- Определение состава оборудования и интеграций.
- Согласование технического задания и критериев приёмки.

2.2. Проектирование

- Проектирование модели данных: зоны, секции, ячейки, PTL, дисплеи, колонны, маршруты, заказы.
- Проектирование очереди операций и алгоритмов распределения нагрузки.
- Проектирование API и интеграционных потоков.
- Проектирование схемы развёртывания и резервного копирования.

2.3. Разработка

- Ведение исходного кода в Git (Gitea).
- Разделение на независимые сервисы: `ascs_api`, `ascs_ui`, `ptl_api`, `rid_api`, `lighthouse_api`.
- Использование стека: Python/FastAPI, React/TypeScript, Go.
- Обязательное ревью изменений и проверка сборки.

- Ведение документации в репозитории и на сайте.

2.4. Тестирование

- Модульные и интеграционные тесты компонентов.
- Проверка API и сценариев через демо-стенд.
- Проверка взаимодействия с эмуляторами PTL, дисплеев и реле.
- Проверка миграций базы данных на чистой базе и на копии рабочей базы.
- Приёмочное тестирование на площадке заказчика.

2.5. Сборка и выпуск

- Сборка Docker-образов и/или бинарников.
- Присвоение версии по SemVer.
- Формирование контрольных сумм SHA256.
- Подпись релиза ключом minisign.
- Публикация на `pkg.whsoft.ru` и в Docker Registry.
- Обновление указателя `latest.txt` и `manifest.json`.

2.6. Внедрение

- Развёртывание в контуре заказчика.
- Настройка зон, секций, раскладки, контроллеров, дисплеев и колонн.
- Интеграция с WMS/ERP заказчика через REST API или CSV.
- Обучение пользователей.
- Опытная эксплуатация и приёмка.

2.7. Эксплуатация

- Работа диспетчера и операторов в АРМ.
- Мониторинг оборудования и очереди.
- Ведение журнала событий.
- Резервное копирование базы и конфигураций.
- Обновление по согласованному графику.

2.8. Сопровождение и модернизация

- Приём обращений по электронной почте.
- Диагностика по логам, метрикам и журналу событий.
- Выпуск исправлений и обновлений.
- Доработка под новые процессы и оборудование.
- Добавление новых модулей и интеграций.

2.9. Вывод из эксплуатации

- Уведомление пользователей о прекращении поддержки версии.
- Предоставление возможности экспорта данных.
- Сохранение резервных копий в течение согласованного срока.
- Отзыв доступов и остановка сервисов.

3. Управление версиями

3.1. Семантическое версионирование

Используется SemVer: `MAJOR.MINOR.PATCH`.

- `MAJOR` — несовместимые изменения API или модели данных;
- `MINOR` — новые функции с обратной совместимостью;
- `PATCH` — исправления без изменения API.

Версия компонента хранится в файле `version` и вшивается в сборку. Для Go-сервисов версия доступна по `GET /version`.

3.2. Ветвление и контроль изменений

- Основная ветка — `master`.
- Разработка ведётся в тематических ветках.
- Изменения проходят ревью и сборку.
- История изменений хранится в Git.
- Релизные теги соответствуют выпущенным версиям.

3.3. Неизменяемость релизов

Опубликованный каталог версии на `pkg.whsoft.ru` не перезаписывается. Для исправления выпускается новая версия. Это обеспечивает воспроизводимость поставки и проверку контрольных сумм.

4. Процесс сборки и выпуска

4.1. Go-сервисы

Для `ptl_api`, `rid_api`, `lighthouse_api` используется скрипт `release.sh`:

1. читает версию из `./version`;
2. собирает бинарник `linux/amd64`;
3. формирует `SHA256SUMS`;
4. подписывает `SHA256SUMS` ключом minisign;
5. формирует `manifest.json`;

6. публикует артефакты в `/var/www/pkg/<product>/<version>/`;
7. обновляет `latest.txt`.

Публикация выполняется по SSH на сервер `pkg.whsoft.ru`. Приватный ключ подписи хранится вне сервера публикации.

4.2. Docker-образы

`ascs_api`, `ascs_ui` и сервисы оборудования собираются в Docker-образы `linux/amd64`. Образы тегируются версией, датой и коммитом. Для проверочного экземпляра образы сохраняются в `images/*.tar` и передаются в offline-архиве.

4.3. Проверка релиза

Перед публикацией выполняется:

- сборка на чистой среде;
- проверка `SHA256SUMS`;
- проверка подписи minisign;
- запуск демо-стенда;
- проверка входа, очереди, маршрута, заказа и оборудования;
- проверка миграций на чистой базе.

5. Установка и обновление

5.1. Установка

- Offline: `install.sh --offline` загружает образы и запускает Compose.
- Online: `install.sh` собирает образы из исходников.
- Конфигурации вынесены в `configs/`, данные — в Docker-тома.
- Демо-данные загружаются скриптом `seed/seed_demo.py`.

5.2. Обновление

1. Резервное копирование базы и конфигураций.
2. Остановка Комплекса.
3. Загрузка нового архива/образов.
4. Запуск `install.sh`.
5. Автоматическое применение миграций.
6. Проверка health-check и основных сценариев.

Конфигурации и данные сохраняются. При несовместимых изменениях выпускается мажорная версия и инструкция по миграции.

5.3. Откат

- Остановка Комплекса.
- Восстановление предыдущих образов.
- Восстановление резервной копии базы.
- Запуск предыдущей версии.
- Проверка работоспособности.

6. Техническая поддержка

6.1. Каналы

- Электронная почта: `support@whsoft.ru`.
- Общие вопросы: `info@whsoft.ru`.
- Документация: `https://whsoft.ru/docs/ascs/`.
- Сайт: `https://whsoft.ru/`.

6.2. Порядок работы

1. Пользователь направляет обращение с описанием проблемы, версией, логами и снимками экрана.
2. Специалист поддержки классифицирует обращение и воспроизводит проблему.
3. При необходимости запрашиваются дополнительные данные.
4. Исправление выпускается в очередной версии или предоставляется обходное решение.
5. Пользователь подтверждает устранение проблемы.

6.3. Уровни критичности

Уровень	Описание	Приоритет
Критический	Комплекс недоступен, данные теряются, оборудование неуправляемо	наивысший
Высокий	Нарушен основной сценарий наборки или печати	высокий
Средний	Ошибка в отдельной функции, есть обходной путь	обычный
Низкий	Косметическая ошибка, пожелание	плановый

7. Резервное копирование и восстановление

- Ежедневный дамп PostgreSQL.
- Копирование конфигураций `configs/` и `.env`.
- Копирование журналов и печатных шаблонов.
- Хранение резервных копий вне сервера приложения.

- Периодическая проверка восстановления на тестовом стенде.
- Срок хранения — не менее 30 дней (уточняется регламентом заказчика).

Пример резервного копирования:

```
docker compose exec -T db pg_dump -U aks aks > backup_$(date +%F).sql
tar -czf configs_$(date +%F).tar.gz configs/
```

8. Безопасность

- Пароли хранятся в виде bcrypt-хэшей.
- JWT-токены имеют ограниченный срок действия.
- Секреты (JWT, пароли БД, ключи) не хранятся в Git.
- Рекомендуется размещение за reverse proxy с TLS.
- Рекомендуется ограничение сетевого доступа к API и БД.
- Обновления зависимостей выполняются по мере выхода исправлений безопасности.
- Телеметрия обезличена и может быть отключена.

9. Телеметрия

Модули оборудования могут отправлять обезличенную телеметрию:

- версия и ОС;
- количество контроллеров, дисплеев, реле и модулей;
- счётчики использования функций;
- базовая надёжность и количество ошибок.

Не передаются: IP-адреса, MAC-адреса, серийные номера, содержимое экранов, персональные данные, учётные данные. Телеметрия отключается настройкой `telemetry_enabled=false` или переменной окружения.

10. Сроки поддержки и вывод из эксплуатации

- Плановая поддержка текущей мажорной версии — не менее 24 месяцев с даты выпуска.
- Критические исправления безопасности выпускаются вне зависимости от плана.
- Уведомление о прекращении поддержки — не менее чем за 6 месяцев.
- После прекращения поддержки пользователю предоставляется возможность экспорта данных и перехода на актуальную версию.
- Резервные копии хранятся в течение согласованного срока, но не менее 30 дней после вывода из эксплуатации.

11. Роли и ответственность

Роль	Ответственность
Правообладатель	Развитие продукта, выпуск версий, лицензирование, поддержка
Разработчик	Реализация функций, исправление ошибок, сборка релизов
Тестировщик	Проверка функций, миграций и совместимости
Специалист поддержки	Приём обращений, диагностика, консультации
Интегратор	Внедрение, настройка оборудования, обучение пользователей
Заказчик	Эксплуатация, резервное копирование, соблюдение регламентов