

Документация о хранении исходного/объектного кода и компиляции

1. Общие сведения о проекте

- **Наименование ПО:** TexX (настольное JavaFX-приложение для управления роботизированным комплексом).
- **Разработчик:** АО "РТ-Техприемка".
- **Используемый стек:** Java 17, JavaFX, Maven.

2. Хранение исходного кода

- **Система контроля версий:** Git. Центральный репозиторий размещается во внутреннем Git-сервере организации (Gitflic), доступ предоставляется по SSH по спискам сотрудников.
- **Структура репозитория:**
- `pom.xml` — описание зависимостей и сборочного процесса Maven.
- `src/main/java` — исходные коды приложения (контроллеры UI, сервисы интеграции, модели уведомлений). Основные классы: `TexXApplication`, `DeviceDialogController`, `DeviceControlController`, `RobotClient`, `Toast`.
- `src/main/resources` — ресурсы интерфейса (FXML-макеты, таблицы стилей, файлы локализации, графические ресурсы, конфигурация логирования).
- **Политика ветвления:**
- `main` — стабильная ветка, содержащая проверенные релизы.
- `develop` — ветка интеграции для подготовки релизов.
- Функциональные ветки формата `feature/<описание>`.
- **Резервное копирование:**

- Ежедневные снапшоты репозитория на резервный сервер (S3-совместимое хранилище с версионностью).
- Еженедельные офлайн-копии на защищённый носитель с хранением в огнестойком сейфе.
- **Контроль доступа:**
- Двухфакторная аутентификация на Git-сервере.
- Разграничение ролей: разработчик (push в feature-ветки), тимлид (merge), администратор (управление пользователями).

3. Хранение объектного кода

- **Артефакты сборки:** Uber-JAR `texx.jar` , формируемый Maven Shade Plugin.
- **Хранилище:** Nexus Repository Manager (раздел «Releases» и «Snapshots»). Доступ только для CI/CD и релиз-менеджеров.
- **Политика хранения:**
- Снапшоты — не более 30 дней.
- Релизы — бессрочно, с обязательным указанием контрольной суммы SHA-256 и подписи разработчика.

4. Процесс компиляции и сборки

1. Разработчик оформляет merge-request в ветку `develop` .
2. CI/CD (GitLab Runner на базе отечественной ОС Astra Linux) запускает сценарий:

```
mvn -B clean verify
```

```
mvn -B package
```

3. В ходе сборки Maven использует `maven-compiler-plugin` с параметрами `source=17` , `target=17` (см. `pom.xml`).
4. Плагин `maven-shade-plugin` упаковывает приложение вместе с зависимостями и прописывает главный класс `ru.rttec.MainLauncher` .

5. Итоговый JAR проходит статический анализ (SpotBugs) и антивирусную проверку (внутренний продукт Касперский).
6. После успешной проверки artefact публикуется в Nexus и помечается версией.

5. Управление конфигурацией и секретами

- Конфиденциальные параметры (URL роботизированного комплекса, ключи API) не входят в репозиторий. Они передаются через переменные окружения и шаблоны `application.properties`.
- Для разработки используется локальный файл `application-dev.properties`, для эксплуатации – параметры окружения в сервисах деплоя.

6. Обеспечение целостности и прослеживаемости

- Для каждого релиза формируется запись в журнале поставок (номер версии, дата, состав изменений, ответственный).
- Все коммиты подписываются GPG-ключами разработчиков.
- Проверка целостности артефактов осуществляется путем сверки SHA-256 перед деплоем.