5. Техническая спецификация хранения кода программного обеспечения

Содержание

1. Введение

1.1. Назначение документа

Данный документ описывает технические средства хранения исходного кода, объектного кода и процессы компиляции программного обеспечения BusStory. Система включает компоненты для edge-устройств и веб-интерфейса с различными требованиями к разработке и развертыванию.

1.2. Архитектура системы хранения

Распределенная система управления кодом:

- Централизованные репозитории для исходного кода
- Системы автоматизированной сборки и тестирования
- Артефакт-репозитории для готовых к развертыванию компонентов
- Системы управления конфигурациями и зависимостями

2. Системы хранения исходного кода

2.1. Система контроля версий

Основная система управления кодом:

- Распределенная система контроля версий (Git)
- Централизованный хостинг с географическим резервированием
- Стратегия ветвления: feature branches с integration workflow
- Автоматизированные проверки качества кода при каждом изменении

Структура репозиториев:

```
main (production)
├── develop (интеграционная ветка)
├── feature/* (разработка новых функций)
├── hotfix/* (критические исправления)
└── release/* (подготовка к релизу)
```

Организация репозиториев:

- **edge-components** - код для устройств подсчета
- **web-interface** - веб-приложение и пользовательский интерфейс
- **shared-libraries** - общие библиотеки и компоненты
- **infrastructure** - конфигурации развертывания и инфраструктуры
- **documentation** - техническая и пользовательская документация

2.2. Контроль качества кода

Автоматизированные проверки:

- Статический анализ кода перед каждым commit
- Автоматическое тестирование при изменениях
- Проверка стиля кодирования и архитектурных соглашений
- Сканирование на уязвимости безопасности

Код-ревью процесс:

- Обязательное рецензирование всех изменений
- Автоматизированные проверки перед merge
- Документирование архитектурных решений
- Контроль соответствия требованиям безопасности

2.3. Резервное копирование и архивирование

Стратегия резервирования:

- Ежедневное резервное копирование всех репозиториев
- Географически распределенные резервные копии
- Долгосрочное архивирование релизных версий (7+ лет)
- Автоматическое тестирование процедур восстановления

3. Технологии и языки программирования

3.1. Edge-компоненты

Основной стек:

- **Системное программирование:** C/C++ для низкоуровневых компонентов
- **Алгоритмы ИИ:** Python 3.8+ с фреймворками машинного обучения
- **Системная интеграция:** Bash/Shell scripting для автоматизации
- **Конфигурация:** YAML/JSON для настроек и метаданных

Специализированные библиотеки:

- Библиотеки компьютерного зрения и обработки изображений
- Фреймворки для нейросетевых вычислений
- Системы межпроцессного взаимодействия
- Драйверы для работы с оборудованием

3.2. Веб-компоненты

Frontend технологии:

- **Основной фреймворк:** Современный JavaScript фреймворк
- **Язык разработки:** TypeScript для типобезопасности
- **Стилизация:** CSS preprocessors и utility-first CSS фреймворки
- **Сборка:** Современные bundlers с оптимизациями

Backend технологии:

- **Серверная среда:** Node.js runtime environment
- **API разработка:** RESTful и GraphQL API
- **База данных:** NoSQL и реляционные СУБД
- **Кэширование:** In-memory кэш и распределенное кэширование

3.3. Управление зависимостями

Package управление:

- **Python:** pip/conda с virtual environments
- **JavaScript/Node.js:** npm/yarn с lockfiles
- **Системные пакеты:** apt/yum package managers
- **Container dependencies:** Docker multi-stage builds

Безопасность зависимостей:

- Автоматическое сканирование уязвимостей
- Контролируемые обновления с тестированием
- Внутренние proxy-репозитории для критических зависимостей
- Аудит лицензий используемых библиотек

4. Процессы сборки и компиляции

4.1. Автоматизированная сборка

CI/CD Pipeline:

```yaml
# Пример workflow сборки
stages:
  - code-quality     # статический анализ и линтинг
  - unit-tests       # unit тестирование
  - integration      # интеграционные тесты
  - security-scan    # сканирование безопасности
  - build            # компиляция и сборка
  - package          # создание артефактов
  - deploy-staging # развертывание на staging
  - e2e-tests      # end-to-end тестирование
  - deploy-prod    # развертывание на production
```

Оптимизации сборки:

- Параллельная сборка независимых компонентов
- Кэширование промежуточных результатов
- Инкрементальная сборка при небольших изменениях
- Условная сборка в зависимости от затронутых модулей

4.2. Компиляция edge-компонентов

Процесс компиляции:

1. **Cross-compilation** для целевых архитектур
2. **Статическая линковка** для минимизации зависимостей
3. **Оптимизация под целевое железо** (ARM, GPU ускорение)
4. **Создание образов** для развертывания на устройствах

Артефакты сборки:

- Исполняемые файлы для различных архитектур
- Системные образы для быстрого развертывания
- Конфигурационные файлы и скрипты установки
- Контейнеры для изолированного выполнения

4.3. Сборка веб-компонентов

Frontend сборка:

1. **Transpilation** TypeScript в JavaScript

2. **Bundling и минификация** для производства
3. **Asset optimization** (изображения, шрифты, SVG)
4. **Code splitting** для оптимальной загрузки

Backend сборка:

1. **Dependency resolution** и установка пакетов
2. **Database migrations** и схемы данных
3. **API documentation** генерация
4. **Container образы** для развертывания

4.4. Тестирование в процессе сборки

Уровни тестирования:

- **Unit тесты:** проверка отдельных компонентов
- **Integration тесты:** взаимодействие между модулями
- **System тесты:** проверка работы системы целиком
- **Performance тесты:** нагрузочное тестирование
- **Security тесты:** проверка на уязвимости

Критерии качества:

- Покрытие кода тестами: >80%
- Все критические пути должны быть протестированы
- Производительность не должна деградировать
- Отсутствие критических уязвимостей безопасности

5. Управление версиями и релизами

5.1. Семантическое версионирование

Схема версионирования:

- **MAJOR.MINOR.PATCH** (например, 2.1.3)
- **MAJOR:** Breaking changes, несовместимые изменения API
- **MINOR:** Новая функциональность с обратной совместимостью
- **PATCH:** Исправления ошибок и мелкие улучшения

Специальные версии:

- **Alpha/Beta:** Предварительные версии для тестирования
- **Release Candidates:** Кандидаты на релиз
- **Hotfix:** Экстренные исправления критических ошибок

5.2. Процесс релиза

Подготовка к релизу:

1. Создание release branch из develop
2. Финализация функциональности и bug fixing
3. Обновление документации и changelog
4. Комплексное тестирование на staging среде
5. Подготовка deployment скриптов

Выпуск релиза:

1. Merge release branch в main
2. Создание git tag с версией
3. Автоматическая сборка production артефактов
4. Развертывание на production среде
5. Мониторинг стабильности после развертывания

5.3. Управление артефактами

Типы артефактов:

- **Source code archives:** исходный код релизных версий
- **Binary packages:** скомпилированные исполняемые файлы
- **Container images:** Docker образы для развертывания
- **Installation packages:** инсталляторы для различных платформ

Хранение артефактов:

- Centralized artifact repository с версионированием
- Географически распределенные CDN для быстрого доступа
- Цифровые подписи для обеспечения целостности
- Автоматическая очистка устаревших версий

6. Инфраструктура разработки

6.1. Среды разработки

Development окружения:

- Локальная разработка с Docker containerization
- Общие development серверы для интеграционного тестирования
- Feature preview окружения для демонстрации изменений
- Performance testing среды для нагрузочного тестирования

Staging окружения:

- Полная копия production конфигурации
- Автоматическое развертывание из release branches
- Comprehensive testing suite выполнение
- User acceptance testing среда

6.2. Автоматизация DevOps

Infrastructure as Code:

- Версионированные конфигурации инфраструктуры
- Автоматическое провижионирование ресурсов
- Конфигурация через декларативные манифесты
- Rollback capabilities для быстрого отката

Мониторинг и логирование:

- Централизованное логирование всех компонентов
- Метрики производительности и доступности
- Alerting системы для критических событий
- Distributed tracing для диагностики проблем

6.3. Безопасность разработки

Secure Development Lifecycle:

- Security review на каждом этапе разработки
- Automated vulnerability scanning в CI/CD
- Dependency security monitoring
- Regular security training для команды разработки

Access Control:

- Multi-factor authentication для всех систем
- Role-based access control с минимальными привилегиями
- Regular access reviews и ротация ключей
- Audit logging всех действий с кодом

7. Безопасность и контроль доступа

7.1. Защита исходного кода

Контроль доступа к репозиториям:

- Аутентификация через корпоративные системы
- Роле-основанные права доступа (read/write/admin)
- Двухфакторная аутентификация для критических операций
- Аудит всех операций с исходным кодом

Шифрование и защита:

- Шифрование репозиториев в состоянии покоя
- TLS encryption для всех сетевых операций
- Цифровые подписи критических commit'ов
- Защищенные ключи для автоматизированных процессов

7.2. Compliance и аудит

Документирование изменений:

- Детальные commit messages с обоснованием изменений
- Связывание изменений кода с бизнес-требованиями
- Audit trail для всех operations
- Regular compliance audits внешними аудиторами

Retention policies:

- 7+ лет хранения релизных версий
- Архивирование development истории
- Secure deletion устаревших данных
- Backup verification процедуры

7.3. Управление секретами

Хранение конфиденциальной информации:

- Централизованное управление секретами
- Ротация ключей и паролей по расписанию
- Шифрование секретов в конфигурациях
- Отделение секретов от исходного кода

Доступ к production секретам:

- Ограниченный круг лиц с доступом
- Temporary access tokens для deployment
- Мониторинг использования секретов
- Автоматическое revocation при подозрительной активности

Контактная информация:

ООО «ДонНовоТех»
344000, Россия, г. Ростов-на-Дону, ул. М.Горького 205
Генеральный директор: Евгений Львович Левитин
Телефон: +7 904 500 6087
Email: leojohn@yandex.ru
Веб-сайт: https://donnovotech.ru