1. Актуальность темы

Современные пользователи нуждаются в удобных, надёжных и безопасных средствах резервного копирования данных. Многие существующие решения:

- работают только на одной платформе
- требуют сложной настройки
- или предполагают платную подписку

При этом отсутствует простое кроссплатформенное решение с открытой архитектурой, которое бы:

- выполняло корректные бэкапы
- сохраняло их на удалённом сервере
- обладало способностью к масштабированию

Это и определяет актуальность разработки собственной системы резервного копирования.

2. Цель и задачи

Цель – разработать кроссплатформенное клиент-серверное приложение в экосистеме Java, которое обеспечивает резервное копирование и восстановление данных с возможностью хранения на удалённом сервере.

Для достижения цели были поставлены задачи:

- изучить принципы резервного копирования и восстановления данных
- проанализировать существующие архитектуры и средства автоматизации
- сформулировать требования к будущему приложению
- спроектировать архитектуру и реализовать прототип системы
- провести тестирование и проанализировать результаты

3. Анализ решений и проектирование архитектуры

В первой главе ВКР рассмотрены:

- базовые принципы резервного копирования: полное, дифференциальное, инкрементальное копирование, концепция snapshot и Continuous Data Protection (CDP).
- главные приколы: deduplication, шифрование, контроль целостности

- Архитектурные подходы: клиент-сервер, agentless, микросервисные
- инструменты и утилиты: rsync, Restic, Borg, Rclone, Syncthing, Bacula и др.
- примеры хранилищ: локальные диски, NAS, облачные S3, MinIO.

На основе анализа выбраны:

- **Restic** — для создание бэкапов
- **Rclone** — для передачи данных
- **MinIO** — как self-hosted облачное хранилище
- **PostgreSQL** — для хранения пользователей и данных

Разработана архитектура приложения, включающая:

- серверную часть для всего подряд
- клиентскую часть для запуска бэкапов и локальных настроек
- REST-интерфейс для взаимодействия между ними

4. Разработка серверного приложения и инфраструктуры

Сервер написан на Java с использованием Spring Boot.

Реализованы:

- API для регистрации пользователей, настройки расписания, мониторинга состояния;
- база данных PostgreSQL для хранения пользователей и их данных;
- интеграция с Restic и MinIO через системные вызовы и HTTP-запросы;
- возможность масштабирования и изоляции задач с помощью Docker.

Инфраструктура развёрнута на удаленном сервере и протестирована в контейнерах.

5. Разработка клиентского приложения

Клиентское приложение также реализовано на Java. :

- сбор конфигурации и подготовку к бэкапу;
- создание снэпшотов с помощью Restic;
- передачу архивов на MinIO через Rclone;

- восстановление данных по запросу пользователя;
- ведение логов и передачу статуса на сервер.

Все действия автоматизируются с помощью планировщика задач, встроенного в Spring Framework.

6. Результаты работы

Разработано и протестировано в ручном режиме:

- клиент-серверная архитектура;
- резервное копирование и восстановление данных;
- поддержка нескольких платформ (кроссплатформенность через JVM);
- данные хранятся в зашифрованном виде;
- приложение работает даже в изолированных условиях.

7. Выводы

Доказали, что с использованием современных open-source решений возможно создать удобную и безопасную систему резервного копирования:

- с кроссплатформенной архитектурой;
- с минимальной нагрузкой на пользователя;
- с возможностью масштабирования и интеграции в CI/CD-среду.

Наше решение может быть применено как в домашних условиях, так и в организациях.