МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА ПРИЛОЖЕНИЯ ОБЛАЧНОГО ХРАНИЛИЩА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Наумова Никиты Игоревича

Научный руководитель
ст. преподаватель _____ Н. Е. Тимофеева

Заведующий кафедрой
к.ф.-м.н., доцент _____ Л. Б. Тяпаев

Саратов 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. В современном мире объем генерируемых данных стремительно увеличивается, что обусловлено цифровизацией, развитием интернета вещей и ростом числа пользователей, работающих удаленно. Облачные хранилища стали ключевым элементом цифровой инфраструктуры, обеспечивая доступ к данным с различных устройств, автоматическое резервное копирование и гибкость в управлении информацией. Однако существующие коммерческие решения, такие как Amazon S3, Google Cloud Storage и Microsoft Azure Blob Storage, часто ориентированы на крупные предприятия и могут быть избыточными или дорогостоящими для небольших команд, стартапов или образовательных проектов. Кроме того, вопросы безопасности данных, зависимости от интернет-соединения и ограниченной кастомизации под специфические нужды пользователей подчеркивают необходимость разработки специализированных решений. Создание серверной части облачного хранилища, способного сочетать высокую производительность, безопасность и простоту локального развертывания, является актуальной задачей, отвечающей современным требованиям к управлению данными.

При разработке такого приложения особое внимание уделяется архитектуре, обеспечивающей надежное хранение, управление файлами и интеграцию с другими системами. Решение должно быть масштабируемым, поддерживать стандарты безопасности (например, шифрование и аутентификацию) и предоставлять интуитивный интерфейс для взаимодействия с пользователями. Разработка подобного приложения позволяет не только удовлетворить практические потребности, но и внести вклад в изучение современных подходов к созданию облачных технологий.

Цель бакалаврской работы – разработка приложения облачного хранилища на основе фреймворка Nest.js, обеспечивающего безопасное хранение и управление файлами.

Поставленная цель определила следующие задачи:

1. Провести анализ существующих решений облачных хранилищ и их характеристик.
2. Обосновать выбор технологического стека для разработки серверной части приложения.
3. Спроектировать архитектуру серверной части с учетом модульности и

масштабируемости.

4. Реализовать систему аутентификации и авторизации пользователей на основе JWT-токенов.
5. Разработать функционал управления файлами, включая загрузку, скачивание, удаление и предпросмотр.
6. Обеспечить безопасность данных с использованием шифрования и оптимизировать производительность системы.

Методологические основы разработки облачных хранилищ представлены в работах М. Армбруста, А. Фокса, Р. Гриффита, С. Ньюмана, а также в стандартах NIST по безопасности информационных систем. Эти исследования охватывают ключевые аспекты облачных технологий, включая архитектуру микросервисов, виртуализацию и стандарты безопасности, которые легли в основу теоретического анализа и практической реализации проекта.

Теоретическая и практическая значимость бакалаврской работы.
Теоретическая значимость. Работа систематизирует знания о современных подходах к созданию облачных хранилищ, включая анализ архитектурных решений, серверных технологий и методов обеспечения безопасности. Проведено сравнение таких решений, как Amazon S3, Microsoft Azure Blob Storage и Google Cloud Storage, что позволило выделить их преимущества (масштабируемость, интеграция с аналитическими инструментами) и недостатки (высокая стоимость, сложность настройки для небольших проектов). Теоретический анализ дополнен изучением стандартов безопасности NIST и принципов проектирования микросервисных приложений, что сформировало основу для выбора технологий и архитектуры.

Практическая значимость. Разработанное приложение представляет собой полноценный программный продукт, пригодный для использования в небольших командах, стартапах и образовательных проектах. Оно поддерживает локальное развертывание, что снижает затраты и повышает контроль над данными. Модульная архитектура на основе Nest.js и использование REST API обеспечивают гибкость и возможность интеграции с другими системами. Приложение предоставляет функционал для регистрации, аутентификации, управления файлами и их безопасного хранения, что делает его конкурентоспособным решением для организаций с ограниченным бюджетом или специфическими требованиями к конфиденциальности.

Структура и объём работы. Бакалаврская работа состоит из введения, четырех разделов, заключения, списка использованных источников и девяти приложений. Общий объем работы – 58 страниц, из них 40 страниц – основное содержание, включая 6 рисунков и 0 таблиц, цифровой носитель в качестве приложения, список использованных источников информации – 20 наименований.

1 КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

1.1 Первый раздел «Обзор приложения и его аналоги»

Первый раздел посвящен изучению облачных хранилищ, их исторического развития, современных аналогов и характеристик разрабатываемого приложения. Облачные хранилища представляют собой системы, обеспечивающие удаленное хранение данных с доступом через интернет. Основные преимущества включают доступность файлов с любого устройства, автоматическое резервное копирование и гибкость управления. Однако существуют вызовы, такие как зависимость от интернет-соединения, риски утечки данных и высокая стоимость коммерческих сервисов при обработке больших объемов информации.

Облачные технологии начали развиваться с появлением сетевых файловых систем, таких как NFS (1984), которые обеспечивали доступ к данным через локальные сети. Современные решения, такие как Amazon S3 (2006), Microsoft Azure Blob Storage и Google Cloud Storage, ввели новые стандарты, включая масштабируемость, интеграцию с аналитическими инструментами и гибкие классы хранения (например, "холодное"и "горячее"хранение). Эти системы ориентированы на крупные предприятия, что делает их менее подходящими для небольших проектов из-за высокой стоимости и сложности настройки.

Разрабатываемое приложение, реализованное на фреймворке Nest.js, ориентировано на небольшие команды и проекты. Оно поддерживает локальное развертывание, что снижает зависимость от внешних провайдеров и повышает контроль над данными. Приложение предоставляет функционал регистрации, аутентификации, загрузки, скачивания и удаления файлов, а также управления доступом с использованием JWT-токенов и шифрования AES. Сравнительный анализ показал, что приложение выгодно отличается простотой настройки, низкой стоимостью эксплуатации и модульной архитектурой, позволяющей легко добавлять новые функции, такие как версионирование файлов или интеграция с внешними сервисами.

Выводы по первому разделу. Анализ существующих решений выявил их ориентацию на крупные предприятия, тогда как разрабатываемое приложение предлагает доступное решение для небольших проектов. Оно обеспечивает баланс между функциональностью, безопасностью и производительностью, делая его подходящим для локального развертывания и адаптации под специфические нужды.

1.2 Второй раздел «Обзор технологии приложения»

Второй раздел посвящен технологическим основам разработки приложения. Рассмотрены ключевые аспекты облачных технологий, принципы работы хранилищ и выбранный технологический стек. Облачные технологии базируются на виртуализации, которая позволяет эффективно использовать вычислительные ресурсы, и моделях обслуживания: IaaS (инфраструктура как услуга), PaaS (платформа как услуга) и SaaS (программное обеспечение как услуга). Облачные хранилища используют клиент-серверную архитектуру с репликацией данных, кэшированием и шифрованием для обеспечения надежности и безопасности.

Серверные технологии, такие как Node.js, обеспечивают асинхронную обработку запросов, что повышает производительность при работе с большим числом пользователей. Микросервисная архитектура, описанная в работах С. Ньюмана, позволяет разделить приложение на независимые модули, упрощая масштабирование и сопровождение. В качестве основного фреймворка выбран Nest.js, работающий на платформе Node.js с использованием TypeScript. TypeScript обеспечивает строгую типизацию, снижая вероятность ошибок, а модульная структура Nest.js упрощает разработку и тестирование. Для работы с базой данных использован PostgreSQL с ORM TypeORM, что обеспечивает удобное управление данными. Аутентификация реализована с использованием JWT-токенов, а данные защищены шифрованием AES-256. REST API обеспечивает стандартизированное взаимодействие с клиентскими приложениями, поддерживая такие операции, как загрузка файлов до 5 МБ, их предпросмотр и множественное скачивание в формате ZIP.

Студентом самостоятельно разработана серверная часть приложения, включающая модули для аутентификации, управления файлами и обработки запросов. Реализованы маршруты REST API для регистрации, авторизации, загрузки, скачивания и удаления файлов, а также проверки прав доступа. Клиентская часть, построенная на Next.js с использованием библиотеки Ant Design, предоставляет адаптивный интерфейс с поддержкой drag-and-drop, пагинации и отображения уведомлений об ошибках. Интеграция клиентской и серверной частей осуществляется через REST API, что обеспечивает стабильность и масштабируемость.

Выводы по второму разделу. Выбор технологического стека, включа-

ющего Nest.js, TypeScript, PostgreSQL и TypeORM, обоснован их способностью обеспечивать производительность, модульность и безопасность. Реализация серверной и клиентской частей демонстрирует высокую степень самостоятельности студента в разработке программного продукта.

1.3 Третий раздел «Структура приложения»

Третий раздел описывает архитектуру приложения, включающую клиентскую и серверную части. Серверная часть, реализованная на Nest.js, состоит из модулей для аутентификации, управления файлами и обработки запросов. Модуль аутентификации использует JWT-токены для проверки пользователей, а модуль управления файлами поддерживает операции загрузки, скачивания, удаления и предпросмотра. Для хранения данных используется PostgreSQL с TypeORM, что обеспечивает надежное управление метаданными файлов и учетными записями пользователей. Архитектура основана на принципах REST API, что гарантирует совместимость с различными клиентами, включая веб- и мобильные приложения.

Клиентская часть, построенная на Next.js с библиотекой Ant Design, включает страницы авторизации, регистрации и панель управления. Панель управления позволяет пользователям просматривать список файлов, загружать новые файлы с помощью drag-and-drop, скачивать файлы (включая множественное скачивание в формате ZIP) и удалять их. Интерфейс адаптирован для различных устройств, а уведомления об ошибках реализованы через компоненты Ant Design. Модульная структура приложения позволяет добавлять новые функции, такие как версионирование файлов или интеграция с внешними сервисами, без значительных изменений в коде.

Выводы по третьему разделу. Разработанная архитектура обеспечивает четкое разделение клиентской и серверной частей, что упрощает сопровождение и масштабирование. Использование REST API и модульной структуры делает приложение гибким и готовым к дальнейшему развитию.

1.4 Четвертый раздел «Разработка приложения»

Четвертый раздел посвящен процессу реализации приложения и достигнутым результатам. Серверная часть разработана на Nest.js с использованием TypeORM для взаимодействия с базой данных PostgreSQL. Реализованы модули для регистрации и аутентификации пользователей с использованием JWT-

токенов, а также для управления файлами, включая загрузку, скачивание, удаление и предпросмотр. Файлы хранятся на сервере с использованием шифрования AES-256, а метаданные (имя, размер, тип) сохраняются в базе данных. Реализованы REST API маршруты для всех операций, включая обработку ошибок и валидацию входных данных.

Клиентская часть, реализованная на Next.js, включает адаптивный интерфейс с формами авторизации и регистрации, а также панель управления с поддержкой drag-and-drop, пагинации и множественного скачивания файлов в формате ZIP. Интеграция клиентской и серверной частей осуществляется через REST API, что обеспечивает стабильное взаимодействие и обработку ошибок. Проведено тестирование, включая unit-тесты для серверных модулей и end-to-end тесты для проверки взаимодействия компонентов, что подтвердило производительность и надежность системы.

Студентом самостоятельно реализованы все ключевые компоненты: серверные модули, API-маршруты, клиентский интерфейс и интеграция через REST API. Приложение поддерживает загрузку файлов до 5 МБ, отображение метаданных и обработку ошибок через уведомления. Модульная архитектура позволяет легко добавлять новые функции, такие как поддержка больших файлов или облачная синхронизация.

Выводы по четвертому разделу. Студентом разработан полноценный программный продукт, включающий серверную и клиентскую части, интегрированные через REST API. Реализованы ключевые функции управления файлами и аутентификации, а модульная архитектура обеспечивает возможности для дальнейшего расширения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была разработана приложение облачного хранилища файлов с использованием современного технологического стека, включающего Nest.js, PostgreSQL и TypeScript. Работа охватывает полный цикл проектирования и разработки программного продукта — от анализа существующих решений до реализации и интеграции клиентской и серверной частей приложения через REST API.

Разработанное приложение обеспечивает безопасное хранение и управление файлами пользователей с поддержкой основных операций: регистрации, аутентификации, загрузки, скачивания и удаления файлов. Безопасность системы достигается посредством использования JWT-токенов, шифрования данных, валидации входной информации и внедрения механизмов контроля доступа.

Проект характеризуется модульной архитектурой, что облегчает масштабирование и дальнейшую модификацию системы. В клиентской части, реализованной на Next.js с применением библиотеки Ant Design, обеспечен интуитивный и адаптивный пользовательский интерфейс, поддерживающий предпросмотр файлов и множественное скачивание в формате ZIP.

Достигнутая модульность, безопасность и производительность делают приложение пригодным для применения в стартапах, небольших командах разработчиков и образовательных проектах.

Таким образом, поставленная цель и задачи достигнуты в полном объёме, а полученные результаты обладают практической значимостью и могут быть использованы в реальных проектах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Google Cloud Storage документация // URL: https://cloud.google.com/storage/docs/ (дата обращения: 10.10.2024).

2 A View of Cloud Computing // URL: https://dl.acm.org/doi/10.1145/1721650.1721672 (дата обращения: 21.10.2024).

3 Chaffey, D. Cloud Computing for Small Businesses: Benefits and Challenges // URL: https://www.smartinsights.com/digital-marketing-platforms/cloud-computing-for-small-businesses/ (дата обращения: 27.11.2024).

4 Fox, A., Griffith, R. Above the Clouds: A Berkeley View of Cloud Computing // URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html (дата обращения: 07.12.2024).

5 Официальная документация Node.js // URL: https://nodejs.org/en/docs/ (дата обращения: 10.01.2025).

6 Официальная документация Nest.js // URL: https://docs.nestjs.com/ (дата обращения: 17.01.2025)