AdsApi Project – Technical Summary

# Overview

AdsApi is a production-grade .NET 8 Minimal API implementing a scalable and fault-tolerant backend for a classified ads platform. It demonstrates best practices for distributed caching, atomic persistence, structured logging, and validation — all containerized with Docker and Redis Stack.

# Key Technologies and Architecture

• Language & Framework: C# (.NET 8) – Minimal API architecture.  
• Persistence: Redis Stack (RedisJSON + Streams) for fast in-memory reads and reliable persistence.  
• Distributed Locking: Redis-based lock (LockTake/LockRelease) ensures single-writer safety across instances.  
• Atomic File Writes: Uses File.Replace to safely persist data from Redis to ads.json.  
• Validation: FluentValidation for strict DTO validation (Create/Update Ad, Comment).  
• Logging: Serilog structured logging with deterministic sampling for INFO logs and full WARN/ERROR retention.  
• Error Handling: Centralized middleware generating ProblemDetails-compliant error responses.  
• Documentation: OpenAPI/Swagger with typed request/response contracts.  
• Containerization: Dockerfile + docker-compose.yml for API and Redis Stack deployment.

# Core Components

• AdService – CRUD, filtering, sorting, pagination, and geo-radius search.  
• CommentService – Add/List/Delete operations for comments per ad.  
• PhotoService – Handles image uploads, validation, resizing (ImageSharp), and thumbnail generation.  
• OutboxWriter – Background worker flushing Redis data to ads.json using atomic writes and distributed locks.  
• ErrorHandlingMiddleware – Converts unhandled exceptions to standardized ProblemDetails responses.  
• LoggingExtensions – Configures Serilog structured logging and sampling filter for performance.  
• AdRepository – Manages RedisJSON operations and ensures data integrity.  
• Docker Compose – Launches AdsApi and Redis Stack together for local or production environments.

# Performance and Reliability

• RedisJSON enables millisecond read latency.  
• Redis Streams + OutboxWriter guarantee reliable persistence and event-driven durability.  
• Distributed lock prevents concurrent writes to the ads.json file.  
• Atomic file replacement ensures corruption-free writes.  
• Structured logs and health endpoints support observability and diagnostics.

# API Design and Validation

• FluentValidation enforces validation rules on incoming DTOs.  
• Consistent API contracts with standardized ProblemDetails error objects.  
• Supports conditional ETags (If-Match, If-None-Match) for concurrency safety.  
• Includes pagination headers, metadata objects, and OpenAPI documentation for front-end integration.

# Logging and Monitoring

• Serilog structured JSON logging with scoped properties (operation, adId, durationMs, status).  
• Deterministic sampling filter retains WARN/ERROR while reducing INFO noise.  
• Audit logging wraps all data mutations for full traceability.

# Deployment and Configuration

• Run locally: `dotnet run`  
• Docker Compose: `docker compose up --build`  
• Health endpoint: `/healthz`  
• Swagger UI: `/swagger`  
• Redis connection configurable via Ads\_\_Repository\_\_RedisConnection environment variable.  
• Default Redis image: redis/redis-stack:latest (includes RedisJSON module).