CODEVISTO

CODING TRUTH THAT POWERS DECISIONS.

Security and Data Protection Report

Last updated February 17, 2026

1. Overview

Codevisto is a software development activity analysis platform. It connects with your organization's code repositories to extract technical metadata — commits, pull requests, reviews — and from that data, generates metrics and reports that help your team understand how they work.

All our infrastructure operates within the European Union. The architecture is designed around a clear principle: collect only what is strictly necessary and protect it with the appropriate mechanisms at every layer.

Codevisto does not store source code. Not fragments, not partial copies, not complete repositories. The platform stores exclusively technical metadata and the analysis results it generates.

2. Infrastructure

Codevisto runs on Heroku in its European region, a cloud platform owned by Salesforce with SOC 2 and ISO 27001 certifications. The architecture is based on isolated containers for application processes.

Database and cache services are managed by Heroku, delegating physical infrastructure, networking, virtualization, and system updates to a specialized provider. This model reduces the operational risk surface by keeping these layers under the responsibility of a dedicated team.

3. Encrypted Communications

All traffic between users and the platform travels encrypted via TLS. Certificates are managed and renewed automatically. HTTPS is active on all production domains without exception.

Internal connections between application components also use TLS encryption, so data in transit is protected both externally and within the infrastructure itself.

4. Authentication and Access Control

Each user accesses Codevisto with their own credentials. Passwords are never stored in plain text — they are kept using a cryptographic hashing process with a high computational cost level, making them resistant to brute force attacks.

Access is organized by company. Each user belongs to a single organization and can only view that organization's data. There is no possibility of accessing, by mistake or misconfiguration, another company's information.

The system supports email invitations and password recovery through secure single-use links.

5. Authorization and Organizational Isolation

Beyond authentication, every action within the platform goes through an authorization system based on explicit policies. These policies define what each user can see and do based on their role and company membership.

Every data query automatically applies an organizational filter before execution. This means that isolation between clients does not depend on a developer remembering to add a filter — it is a structural constraint of the architecture itself.

6. Data We Handle and Data We Don't

This is probably the most relevant point for any organization evaluating Codevisto.

Data we store: Commit metadata (identifier, message, date, author, change volume, commit type). Analysis results (effort estimation, complexity, test coverage indicators, ratings). Pull requests and reviews (identifier, number, title, state, dates, comments). Organizational information (company data, teams, developers: name, email, username). Access logs for auditing and security purposes.

Data we do NOT store: Source code. File contents. Complete diffs (processed in memory, discarded upon completion). Repository copies (temporary partial copies are deleted once analysis is complete).

7. Repository Access Credential Protection

To connect with your repositories (GitHub, GitLab, Bitbucket), Codevisto requires an access token provided by your organization. Protecting this token is a priority: it is stored encrypted at rest in our database and only decrypted at the exact moment of use. Encryption keys are separated from the application code. Error messages are automatically sanitized to remove any token or password before they can appear in logs. The logging system excludes any field that might contain sensitive information.

8. Web Attack Protection

The platform includes active protection against Cross-Site Request Forgery (CSRF) attacks. Every form and every request generated from the interface includes a verification token that the server validates before processing any action.

All communications are forced over SSL in production, preventing unencrypted requests.

9. Sensitive Configuration Management

Configuration containing sensitive information (API keys, external service credentials, encryption secrets) is managed through a centralized environment variable system with documented defaults and explicit typing. There are no credentials in the source code or in the application repository.

Application startup in production mandatorily requires a master key. Without it, the environment cannot be deployed, preventing an instance from starting without the correct security configuration.

10. Cached Data

The caching system is designed with deliberate restrictions: only simple, non-sensitive data is cached (numeric identifiers and basic data structures). Complete database records or personal information are never cached. The connection to the cache service is encrypted with TLS. Each entry has a defined expiration policy and invalidation mechanisms that guarantee data consistency.

11. Data Lifecycle

Codevisto has active cleanup mechanisms that allow deleting activity data by repository and time period. These operations cascade-delete all derived information (analyses, exclusions) and coherently update the cache.

Database backups are managed at the infrastructure level with automatic retention policies native to the database service provider.

For our internal development and testing environments, we use an anonymization system that generates database copies with all client information obfuscated, so real data is never used outside the production environment.

12. Controls Summary

Communications: TLS encryption at all points, both external and internal.

Authentication: Cryptographic hashing with high computational cost.

Authorization: Explicit policies with mandatory organizational isolation.

Repository credentials: Encrypted at rest, decrypted only on use.

User passwords: Never stored in plain text.

Logs: Automatic filtering of any sensitive data.

CSRF attacks: Active protection across the entire application.

Cache: Encrypted in transit, non-sensitive data only, controlled expiration.

Source code: Not stored — temporary, limited, ephemeral access only.

Configuration: Centralized, no credentials in code, mandatory master key.

Auditing: Access logs and change traceability.

Non-production environments: Client data anonymized.