Bitwarden Server Project Structure

The server-side backend for the Bitwarden password manager. A highly secure .NET application following Clean Architecture principles.

Updated 2025-12-30

#bitwarden #csharp #dotnet #security #api #identity

PNG PDF Copy Prompt

Project Directory

```
bitwarden-server/
  src/ Main Applicatio…
    Core/ Domain logic & …
    Api/ REST API Contro…
    Identity/ OIDC Identity S…
    Infrastructure.EntityFramework/ Primary Persist…
    Infrastructure.Dapper/ High-performanc…
    Sql/ SQL Migrations …
  test/ Unit & Integrat…
  util/ Helper utilitie…
  bitwarden-server.sln Solution file
  global.json .NET version
```

Repository Info

Repository - bitwarden/server

Stars - 15k+

License - AGPL-3.0

Last Analyzed - December 2025

Tech Stack

Language - C# 12

Framework - .NET 8.0

Identity - IdentityServer4

ORM - EF Core & Dapper

Database - SQL Server / Postgres / MySQL

Architecture - Clean Architecture / N-tier

Architecture Notes

Bitwarden Server follows a robust **Clean Architecture** pattern. The `Core` project contains the domain models and service interfaces, ensuring that the business logic is decoupled from infrastructure details. They employ a dual-persistence strategy: using **Entity Framework Core** for standard CRUD operations and **Dapper** for performance-critical read paths where raw SQL provides better optimization.

Key Directories

src/Core/ - The heart of the application. It defines the business rules, security policies, and the repository interfaces that other layers implement.

src/Identity/ - The authentication hub. It handles OpenID Connect and OAuth 2.0 flows, separating identity management from the standard application API.

src/Infrastructure.EntityFramework/ - The implementation of the data access layer. It uses EF Core to map the domain models to the physical database schema.

Why This Structure?

Bitwarden is an excellent reference for anyone building high-security C# applications. It demonstrates how to manage complex identity requirements and how to scale data access in a multi-tenant environment using modern .NET patterns.