testing-strategy

Testing Strategy

This document describes the testing approach for the Task Manager application, emphasizing Test-Driven Development (TDD), boundaries, and best practices for Clean Architecture and DDD.

---

1. Test-Driven Development (TDD)

- Start by writing failing unit tests for new features or changes.
- Use tests to clarify requirements and drive design decisions.
- Only implement code to make tests pass; refactor as needed.

2. Test Types & Boundaries

Unit Tests

- Target Domain and Application layers.
- Test business logic, invariants, and use case handlers in isolation.
- Use FakeItEasy for mocking dependencies.
- Organize by feature and method (e.g., `tests/TaskManager.UnitTests/Task/CreateTests.cs`).

Integration Tests

- Target Infrastructure and Api layers.
- Test adapters, persistence, and endpoint wiring.
- Use Testcontainers for database and external service integration.
- Organize by feature (e.g., `tests/TaskManager.IntegrationTests/Task/`).

3. Test Organization

- One test class per method for complex types.
- Descriptive test names: `method_underTest_expectedBehavior`.
- Cover invalid scenarios and guard clauses.
- Run tests frequently (`dotnet test` / `dotnet watch`).

4. Example: TDD for Task Creation

1. Write a unit test for `Task.Create(...)` enforcing invariants (e.g., title required).
2. Write a test for `CreateTaskCommandHandler` to ensure it creates and persists a task.
3. Implement only enough code to pass the tests.
4. Add integration tests for the API endpoint (`POST /tasks`).

5. Tools

- **xUnit:** Unit and integration test framework.
- **FakeItEasy:** Mocking dependencies in unit tests.
- **Testcontainers:** Containerized integration tests for databases and services.

6. Best Practices

- Keep tests fast, isolated, and repeatable.
- Prefer first-class collections and value objects in assertions.
- Avoid testing implementation details; focus on behavior and outcomes.
- Use Conventional Commits for test changes (e.g., `test(task): add unit tests for task creation`).

See also: Sample Solution Architecture, Feature Walkthrough