MangoChango – Tech Thursday

Showcasing the Team Weekly Status Application: Empowering Our Teams through Clean Architecture

2024

MangoChango
Talent as a Service

Agenda

- Introduction
- Team Weekly Status Application
- Understanding Our Application's Architecture
- 5 minutes break
- Live Coding Session
- Selling Clean Architecture to Clients as a Consultant
- Surprise
- Final Thoughts

MangoChango
Talent as a Service

Introduction to the Application and Clean Architecture

Our Application:

- A tool used for over a year by our team, Siepe.
- Facilitates weekly reports: accomplishments, plans, roadblocks, PTOs.
- Supports multiple teams with necessary management features.

Why Clean Architecture?

- Aims for maintainability, testability, and flexibility.
- Separates concerns and decouples layers.
- Enables easy adaptation to changes (e.g., swapping databases).

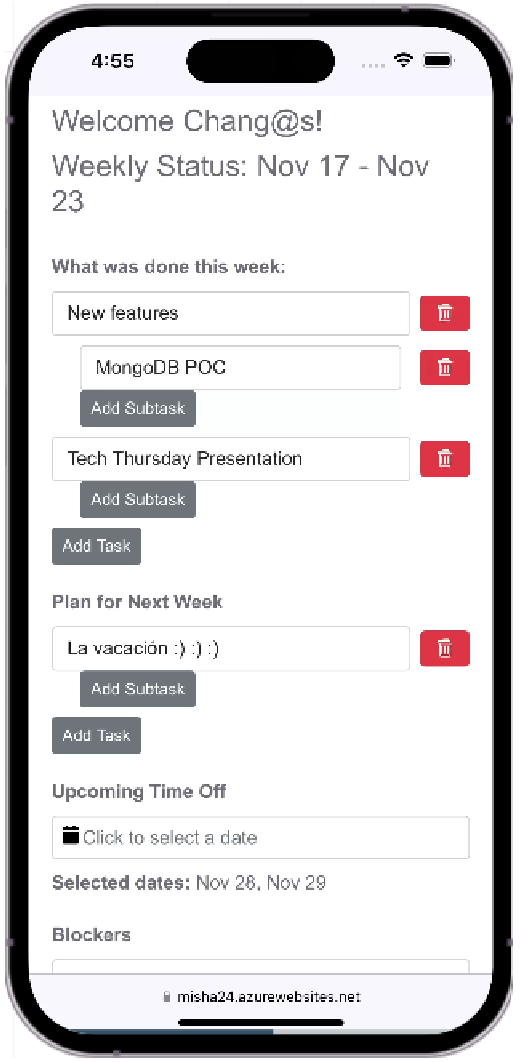
Team Weekly Status Application

Features:

- Facilitates weekly reports: accomplishments, plans, roadblocks, PTOs.
- Multi-Team support with necessary management features.
- Email Reminders: Automated emails to prompt team members to submit their weekly reports.
- Current week reporter automated assignment.
- Ability to copy the report rich-text, and export to PDF and Markdown.
- Authentication with The Jungle credentials.
- Fully responsive.

Upcoming Features:

- **AI-Powered Enhancements:** grammar and fluency improvement.
- **MongoDB** for data persistence.
- **The Jungle Integration**:
  - Seamless access through MangoChango's portal.
  - Single Sign-On (SSO) support for streamlined authentication.


Welcome Chang@s!
Weekly Status: Nov 17 - Nov 23

What was done this week:

New features
MongoDB POC
Tech Thursday Presentation

Plan for Next Week

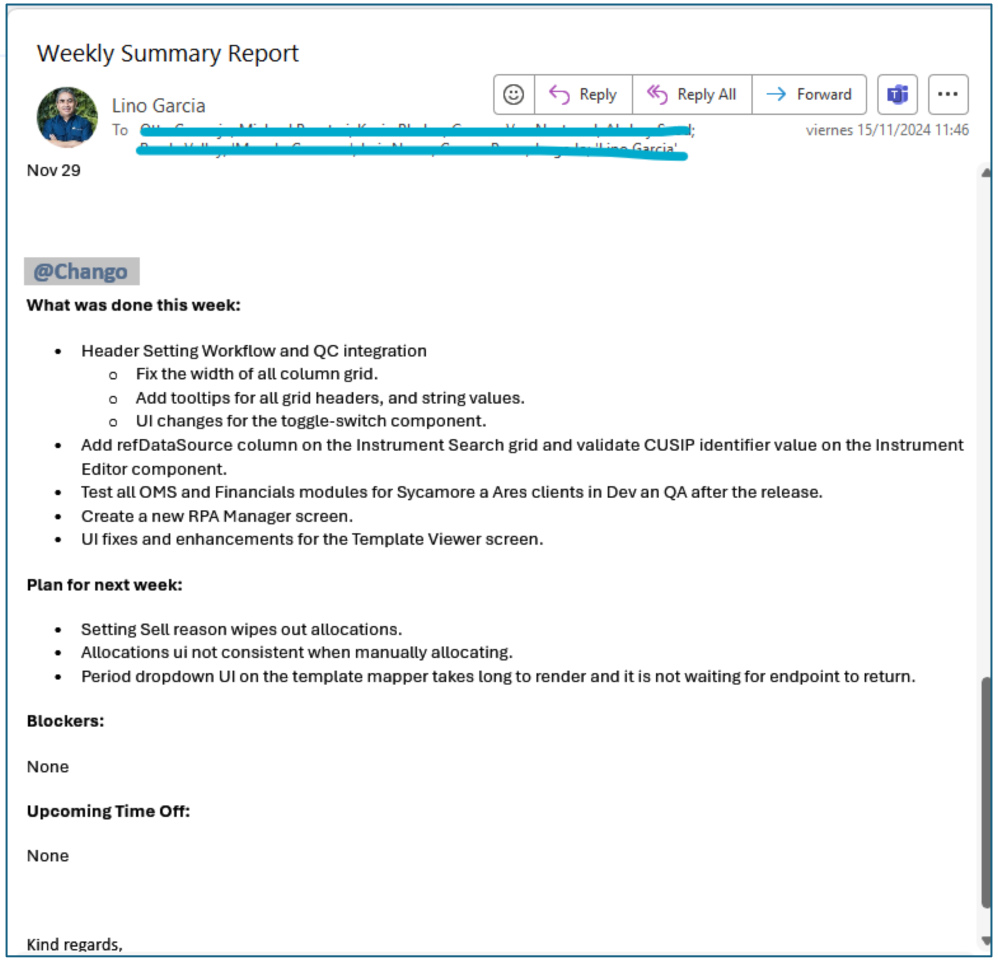
La vacación :) :) :)

Upcoming Time Off

Selected dates: Nov 28, Nov 29

Blockers

Sample email message containing the Weekly Report



Weekly Summary Report

Lino Garcia

To [redacted]

viernes 15/11/2024 11:46

Nov 29

@Chango

What was done this week:

- Header Setting Workflow and QC integration
  - Fix the width of all column grid.
  - Add tooltips for all grid headers, and string values.
  - UI changes for the toggle-switch component.
- Add refDataSource column on the Instrument Search grid and validate CUSIP identifier value on the Instrument Editor component.
- Test all OMS and Financials modules for Sycamore a Ares clients in Dev an QA after the release.
- Create a new RPA Manager screen.
- UI fixes and enhancements for the Template Viewer screen.

Plan for next week:

- Setting Sell reason wipes out allocations.
- Allocations ui not consistent when manually allocating.
- Period dropdown UI on the template mapper takes long to render and it is not waiting for endpoint to return.

Blockers:

None

Upcoming Time Off:

None

Kind regards,

MangoChango
Talent as a Service

It's time for a short demo!

Understanding Our Application's Architecture

Layered Structure:

- **Domain layer:** Core business entities.
- **Application layer:** Business logic.
  - Organized with:
    - **CompositionRoot** for dependency injection.
    - **DTOs, Exceptions, Interfaces, Services.**
  - **Services** like `WeeklyStatusService, ReminderService`.
- **Infrastructure layer**: Data access, external services.
- **WebAPI layer:** Controllers exposing endpoints.
- **ReactJS Frontend:** Consumes APIs, user interface.

Composition Root Pattern:

- Centralizes dependency injection configuration.
- Enhances maintainability and clarity.

Aligning with Clean Architecture Principles

Adopted Principles:

- **Separation of Concerns:** Clear layer boundaries.
- **Dependency Inversion**: Interfaces and Composition Root.
- **Framework Independence:** Core logic decoupled from external tech.

Areas for Improvement:

- **Use Cases in Application Layer**:
  - Currently organized by services, not explicit use cases.
  - Potential to refactor for clearer understanding of application purpose.
- **Value Objects and Domain Events:**
  - Not currently implemented.
  - Could enhance domain modeling and decoupling.
- **Feature-Based Organization** instead of multiple technical layers/projects (maybe?).

Clean vs Hexagonal vs Onion Architecture – Clean Architecture

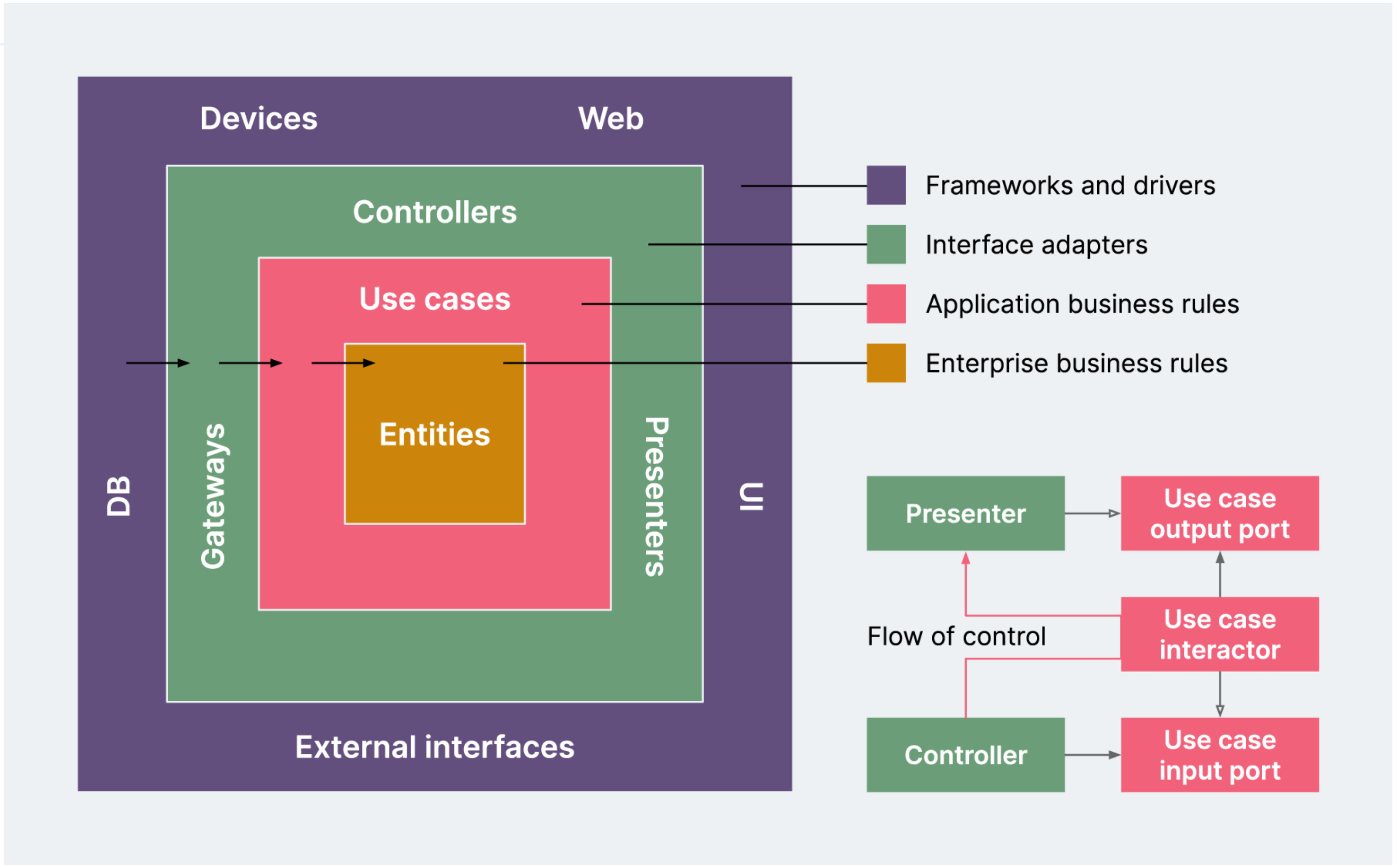


Image source: *https://www.thoughtworks.com/insights/blog/architecture/demystify-software-architecture-patterns*

MangoChango
Talent as a Service

Clean vs Hexagonal vs Onion Architecture – Hexagonal Architecture

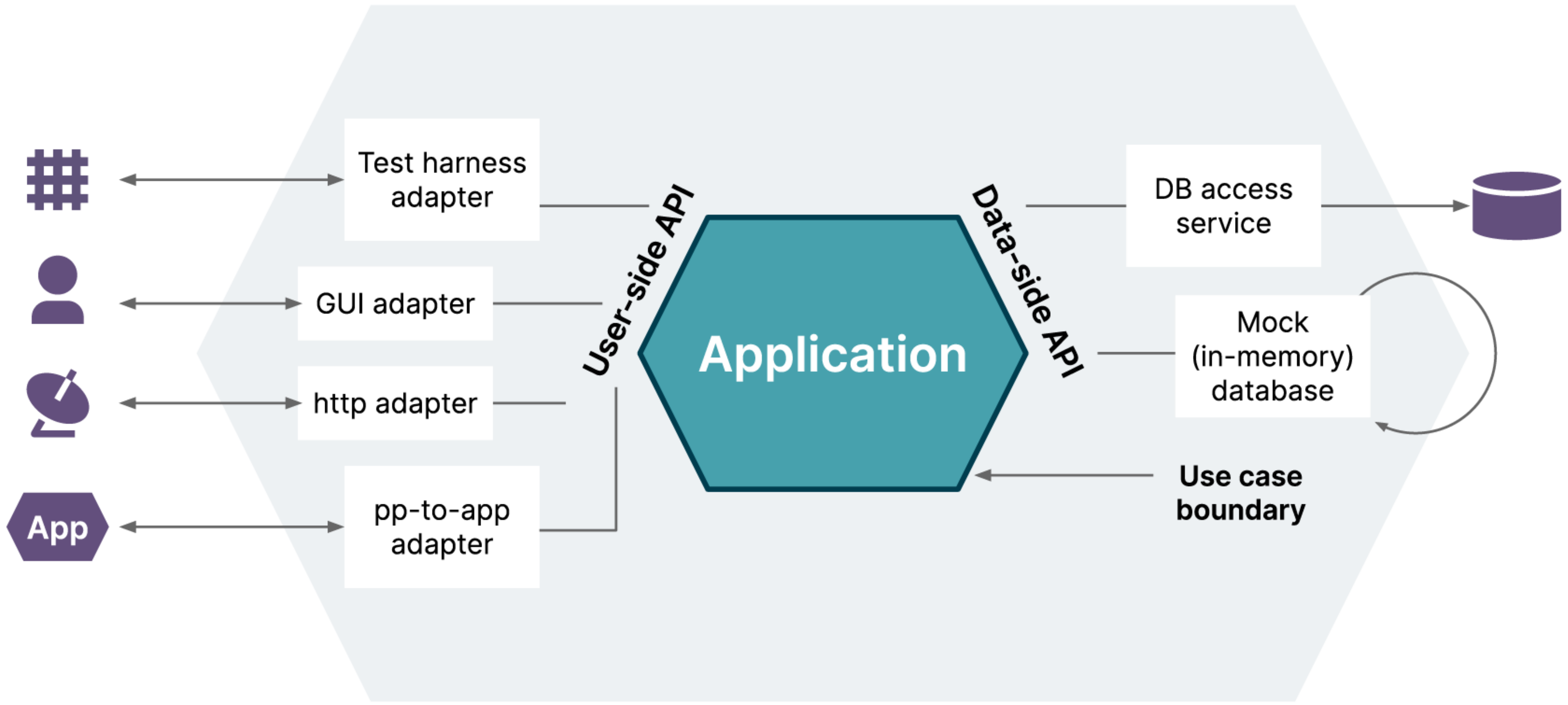


Image source: *https://www.thoughtworks.com/insights/blog/architecture/demystify-software-architecture-patterns*

Clean vs Hexagonal vs Onion Architecture – Onion Architecture

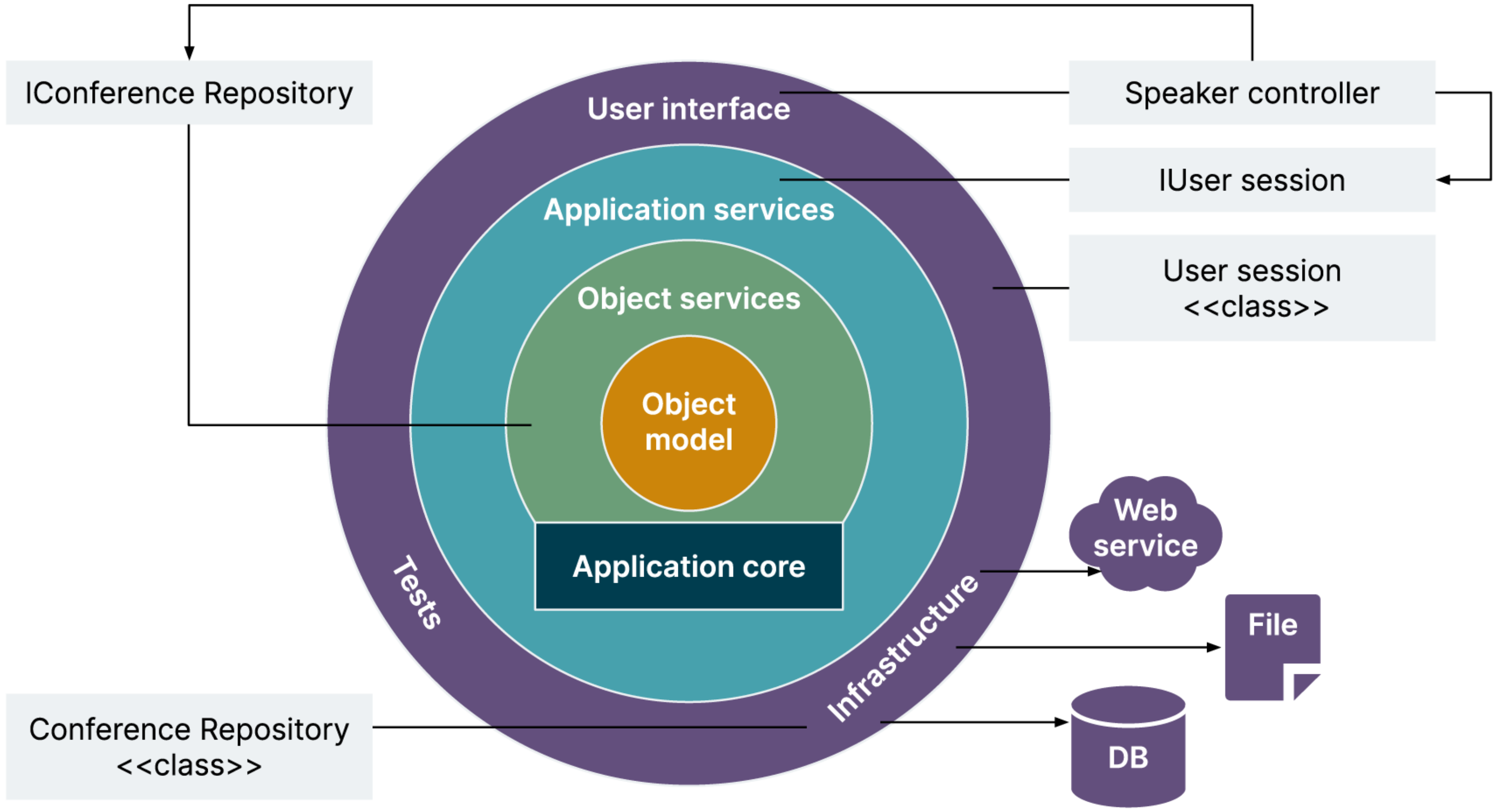


Image source: https://www.thoughtworks.com/insights/blog/architecture/demystify-software-architecture-patterns

It's time for the live code session!

Selling Clean Architecture to Clients as a Consultant

Faster Delivery: Accelerate feature rollout for competitive edge.
Scalability & Flexibility: Easily adapt to growth and changing requirements.
Quality Assurance: Enhance reliability. A cleaner codebase means fewer bugs and a better user experience boosting customer satisfaction.
Technology Agnosticism:

- **Ease of integration:** “You can integrate new technologies or services without overhauling the entire system”.
- **Avoiding Vendor Lock-In:** Flexibility to switch databases or frameworks reduces dependency on specific vendors.

Address Concerns Proactively

- **"Isn't this over-engineering for our needs?"**:
  Proposed response: "The architecture scales with your business. We tailor it to fit your current needs while keeping future growth in mind.".
- **“Will this delay our project?”**:
  Proposed response: "While it might take slightly longer to set up initially, it significantly speeds up future development cycles, leading to earlier overall delivery of features."

Thoughts? Comments? Questions?

Resources

Application repository:

https://github.com/linogvallejo/TeamWeeklyStatusV2

Clean Architecture:

- Robert C. Martin. 2019. Clean Architecture and Design. YouTube.
- Thoughtworks. 2022. Demystifying software architecture patterns, by Rahul Garg.
- Donny Roufs. 2023. Clean Architecture in TypeScript. YouTube.
- Steve Pember. 2023. Anatomy of a Spring Boot App with Clean Architecture by Steve Pember @ Spring I/O 2023. YouTube.
- Tuttodev. Creación de Proyectos con Clean Architecture en NestJS. YouTube.
- Milan Jovanović. 2024. Clean Architecture: The Missing Chapter.

Composition Root:

- DotNetCurry magazine. 2016. Clean Composition Roots with Pure Dependency Injection (DI), by Yacoub Massad.
- Martin Fowler articles. 2023. Dependency Composition, by Daniel Somerfield.

MangoChango

Talent as a Service

www.mangochango.com