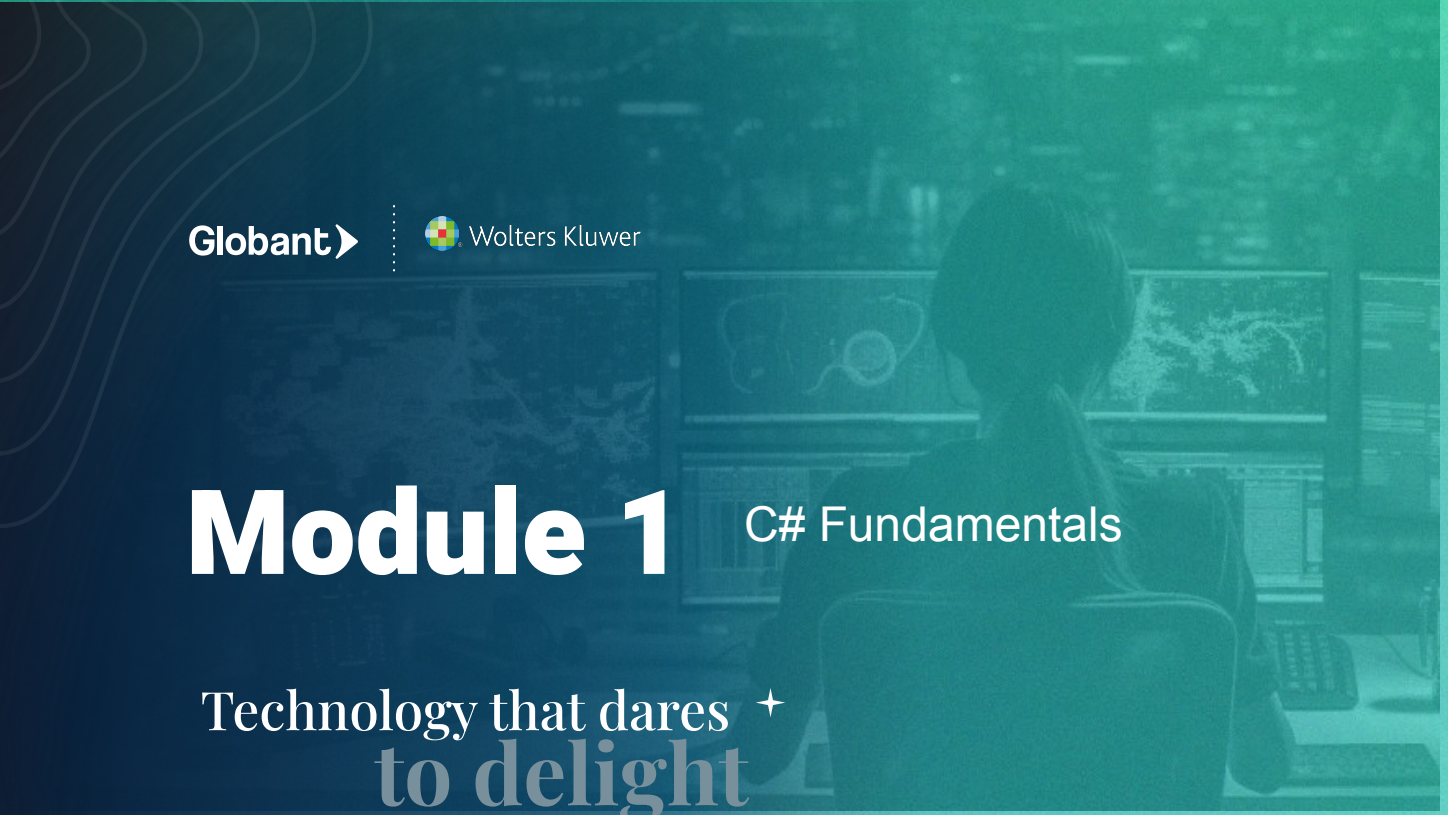
Globant | Wolters Kluwer



Module 1

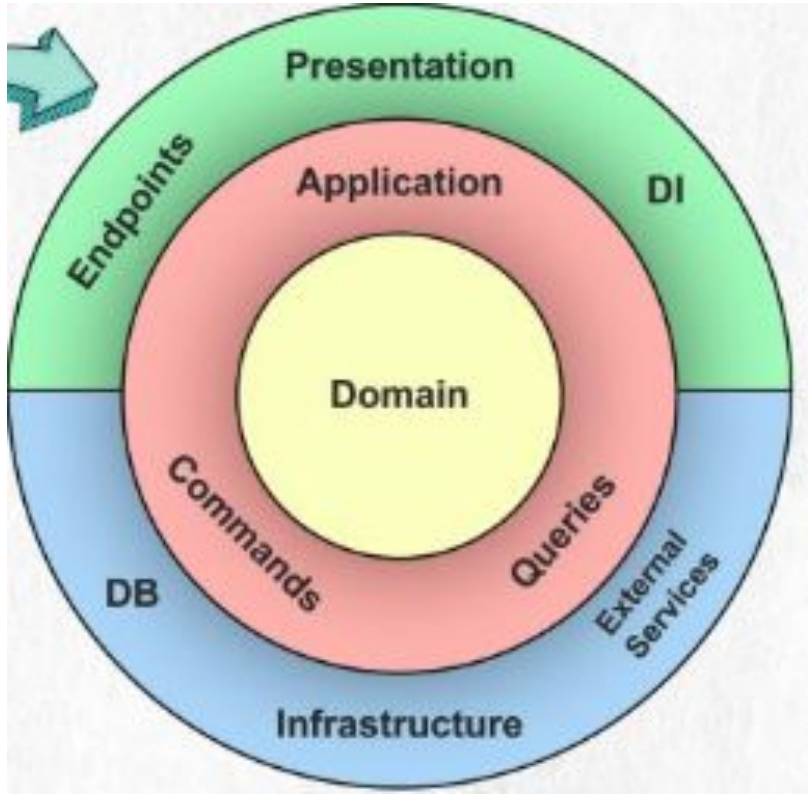
C# Fundamentals

Technology that dares to delight

Why a Todo list?

- Emulate a real case (task management)
- Allow to cover all basic aspects of C#:
  - Variables
  - Data Structures
  - Loops
  - Conditionals
- Address logic, input/output and user flow.
- Github repository: https://github.com/dotnet-bootcamp-2025/MyTodo
-

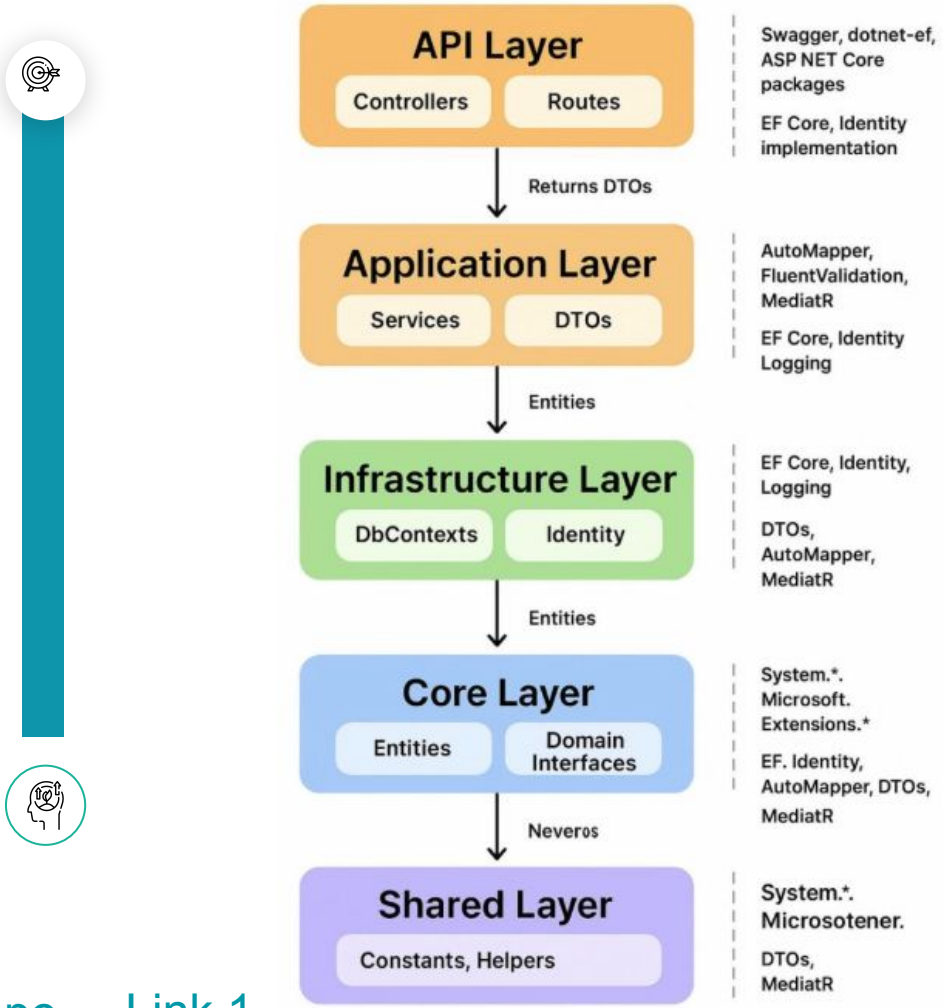
Sneak Peek — Clean Architecture



Repo Link 1

Globant

Sneak Peek — Clean Architecture



Repo Link 1

Globant

Command	Description
`git init`	Initializes a new Git repository in the current directory.
`git clone [repository]`	Creates a copy of an existing Git repository from a remote source.
`git add [file]`	Stages changes in the specified file for the next commit. Use `git add .` to stage all changes.
`git commit -m "[message]"`	Commits the staged changes to the repository with a descriptive message.
`git status`	Displays the status of the working directory and staging area, showing modified, staged, or untracked files.
`git pull`	Fetches changes from a remote repository and merges them into the current branch.
`git push`	Uploads local commits to a remote repository.
`git branch`	Lists all branches in the repository. Use `git branch [branch-name]` to create a new branch.
`git checkout [branch-name]`	Switches to the specified branch. Use `git checkout -b [branch-name]` to create and switch to a new branch.
`git merge [branch-name]`	Merges the specified branch into the current branch.

C# Syntax (Console shell) — m1-01-syntax

Rubric: C# syntax

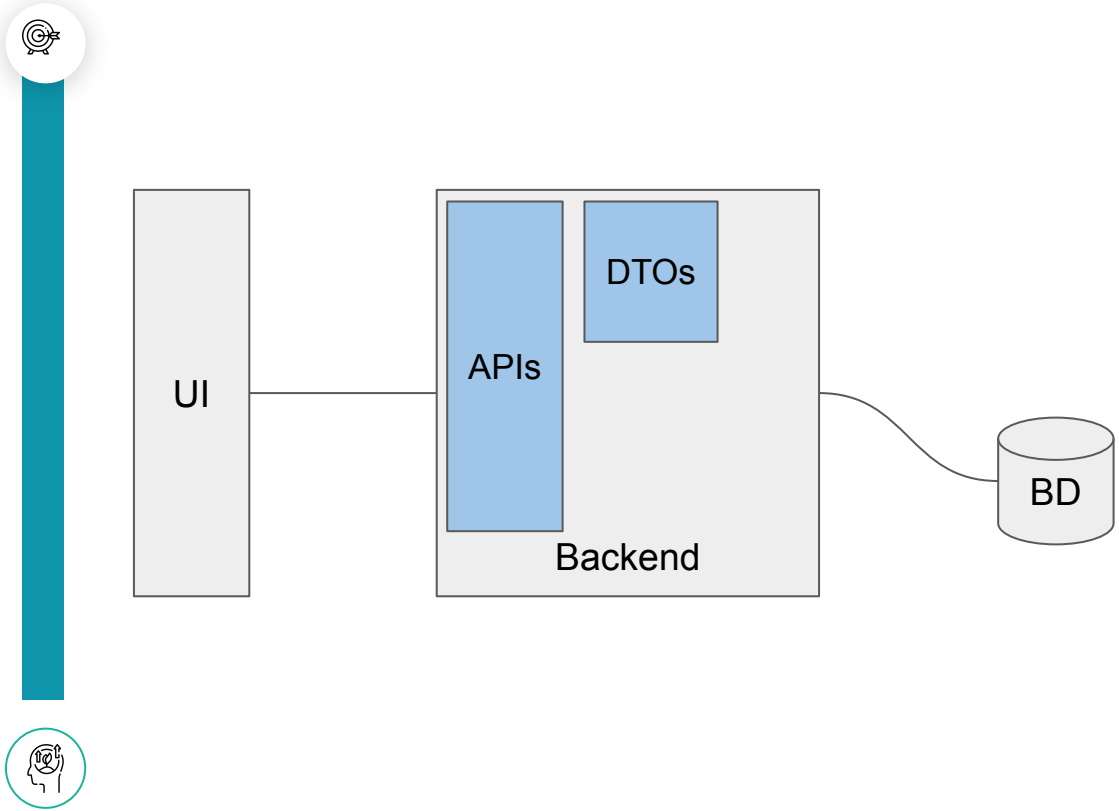
Teach: namespaces, classes/records, methods, top-level program, basic I/O.

Goal: top-level program, namespaces, classes, methods, Main I/O.

- Create src/TodoApp.Console/Program.cs (top-level statements):
- Create the domain model src/TodoApp.Domain/TodoItem.cs

Globant

C# Syntax —
DTOs



Variables in C#: naming, var vs explicit, immutability — m1-02-variables

Rubric: variables (naming, var vs explicit, immutability).

Teach: choose names for intent; var when type is obvious; immutable domain.

Goal: clear names, when to use var, show immutable design.

- In Program.cs, capture inputs with explicit types where clarity matters.
- Initialize the Todo class with the required properties.

Globant

Variables in C#

—

m1-02-variables

- **Espacios en memoria para almacenar datos.**
- **Fuertemente tipadas.**
- **Tipos principales:**
  - **Primitivos (valor):** int, double, bool -> memory - stack
  - **Referencia:** string, object, clases **-> memory - heap**
  - **Implícitos:** var (tipo inferido)

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types

Loops

- Repetir un bloque de código mientras se cumpla una condición.
- Tipos en C#:
  - for
  - foreach
  - while
  - do…while

Data Structures: List<T> first — m1-03-data-structures

Rubric: correct choice (avoid premature optimization).

Teach: simplest structure first; hide behind an interface to allow swaps; avoid premature optimization.

Goal: start simple with List<T>, hide behind an interface so we can swap later.

- Add repository contract src/TodoApp.Domain/ITodoRepository.cs.
- Add in-memory implementation src/TodoApp.Infrastructure/InMemoryTodoRepository.cs.
- Wire a static field in Program.cs for now (later we'll inject)

Globant

Data Structures

—

m1-03-data-structures

Arrays

- **Description:** Fixed-size collection of elements of the same type.
- **Use Case:** When the number of elements is known and does not change.
- **Example:**

```
int[] numbers = { 1, 2, 3, 4, 5 };
```

Lists

- **Description:** Dynamic-sized collection of elements of the same type, implemented using List<T>.
- **Use Case:** When the number of elements can grow or shrink dynamically.
- **Example:**

```
List<string> names = new List<string> { "Alice", "Bob" };
names.Add("Charlie");
```

Dictionaries

- **Description:** Key-value pairs, implemented using Dictionary<TKey, TValue>
- **Use Case:** When you need fast lookups based on unique keys.
- **Example:**

```
Dictionary<int, string> students = new Dictionary<int, string>
{
    { 1, "Alice" },
    { 2, "Bob" }
```

Globant

Data Structures

—

m1-03-data-structures

Globant

HashSet

- **Description:** A collection of unique elements, implemented using HashSet<T>.
- **Use Case:** When you need to store unique items and perform fast lookups.
- **Example:**

```
HashSet<int> uniqueNumbers = new HashSet<int> { 1,  2,  3 };
uniqueNumbers.Add(3); // No duplicates allowed
```

Queues

- **Description:** First-In-First-Out (FIFO) collection, implemented using Queue<T>.
- **Use Case:** When you need to store unique items and perform fast lookups.
- **Example:**

```
Queue<string> tasks = new Queue<string>();
tasks.Enqueue("Task1");
tasks.Enqueue("Task2");
string nextTask = tasks.Dequeue();
```

Stack

- **Description:** Last-In-First-Out (LIFO) collection, implemented using Stack<T>.
- **Use Case:** When elements need to be processed in reverse order of their addition.
- **Example:**

```
Stack<int> stack = new Stack<int>();
```

Data Structures

—

m1-03-data-structures

- **LinkedList**
  - **Description:** TBD.
  - **Use Case:** TBD.
  - **Example:**

    TBD
- **SortedList**
  - **Description:** TBD.
  - **Use Case:** TBD.
  - **Example:**

    TBD
- **ObservableCollection**
  - **Description:** TBD.
  - **Use Case:** TBD.
  - **Example:**

    TBD

- **ConcurrentCollection**
  - **Description:** TBD.
  - **Use Case:** TBD.
  - **Example:**

Globant

Flow Control: loops & conditionals (menu) — m1-04-flow

Rubric: avoid nested complexity; guard clauses; use clear switch.

Goal: sturn menu into real commands, keep functions small.

- Extract small functions.
- Use switch for routing.
- Add Toggle and Delete with guard clauses (no deep nesting).

Clean Code refactor (SRP) — m1-06-clean-code

Rubric: small functions, SRP, comments only when necessary.

Goal: move rules to a service; keep console thin.

- Add src/TodoApp.Application/TodoService.cs.
- Update Console to call the service (parsing/printing only).

Globant

(Optional for semi-seniors) Minimal API — m1-07-api

Rubric: still aligned (clean boundaries, DTOs, proper verbs)

1. src/TodoApp.Api/Program.cs

2. Run & explore Swagger if you've added it, or hit endpoints with curl/Postman

Unit Tests (xUnit) — m1-08-tests

Rubric: clean code (test small units)

1. Sample test tests/TodoApp.Tests/TodoServiceTests.cs:

How to run this in class

Per topic:

- Live demo (10–15 min): explain the concept, paste the minimal snippet, run it.
- Students code (20–30 min): they commit to their current m1-0X-* branch.
- Open a PR → pass CI → quick rubric-driven review.

PR ritual (every step)

- PR name: m1-0X-topic – FirstName LastName.
- Must pass: build/test + PR checklist.