IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

RÉPUBLIQUE FRANÇAISE
*Liberté*
*Égalité*
*Fraternité*

# Introduction to FIAB

Fabien Dagnat

ILSD – FIAB – 0– 2025-2026

## What's the point of distributed systems?

- Performance gains
  - sharing resources
  - sharing computations
- Scaling up
- The system using the software is distributed (*e.g.* a car or a plane)
- Better availability
- Better reliability

## Which difficulties?

- Heterogeneity
- No global clock
- No global state just partial views
- Faults: machine, communication channel
- Security: malicious intent (*e.g.* denial of service, man-in-the-middle)
- Coordinate: reconcile
- Decide: consensus
- Various scale: from Personal Area Network to internet
- ...

## Needs

- Specific data structures
  - distributed
  - replicated
- Specific algorithms
  - Fault tolerant
  - Resistant to arrivals/departures of machines
  - High-performance (with various topologies)

## This teaching unit

- Objectives
  - discover distributed programming and reliability
  - to be able to write distributed programs using the Elixir language
  - become operational by adopting professional practices
- Approach
  - work on a *realistic* e-commerce system
  - supervision by professionals (KBRW)
  - integration of reliability progressively
- Pre-requisite: programming, git
  - Ideally: concurrency

## Organization

1. Elixir
    - Functional aspects S1
    - Concurrency and distribution S2
2. Phase 1, application analysis
    - discovery of business issues, the provided code and the jupyter notebook (S3)
    - Finding defaults and *map/reduce* (S4)
3. Phase 2, GenServer for statistics (S4-6)
4. Phase 3, Transactions and management of errors (S8-9)
5. Phase 4, Distributed architecture (S10-11)

- 5 lectures in parallel on concepts
- Evaluation: 6 homework to be submitted regularly

## Dépôt de ressources et devoirs

- Dépôt git
  - `https://gitlab-df.imt-atlantique.fr/fdagnat/fiab2025-2026`
  - `resources` for resources (mainly the first version of the case study)
  - `homework/`*name* for your personal work
    - directories by homework `hw_`*number*
- µ-tutorial
  - first time: `git clone` *url*
  - then: homework in the corresponding directory `homework/`*name*`/hw_`*number*
  - possibly copying files for the `resources` directory (`cp` ou `cp -R`)
  - adding your files
    - `git add` *files*; `git commit -m "`*a message*`"`; `git push`
    - possibly a *merge*
  - Often *commit* and push regularly

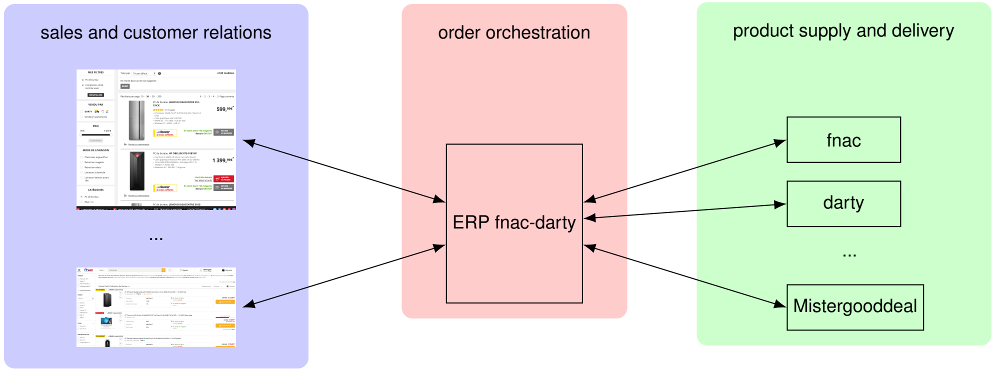
## Elixir

- Functional programming language of the Erlang family
- Proposed by José Valim around 2012
- General principles
  - Functional
  - Concurrent (process executing concurrently)
  - Distributed (notion of node)
  - Dynamic typing (just before execution)
  - Compile to the Erlang Virtual Machine (efficient, distributed, fault tolerant, numerous librairies)

`https://p4s.enstb.org/elixir`

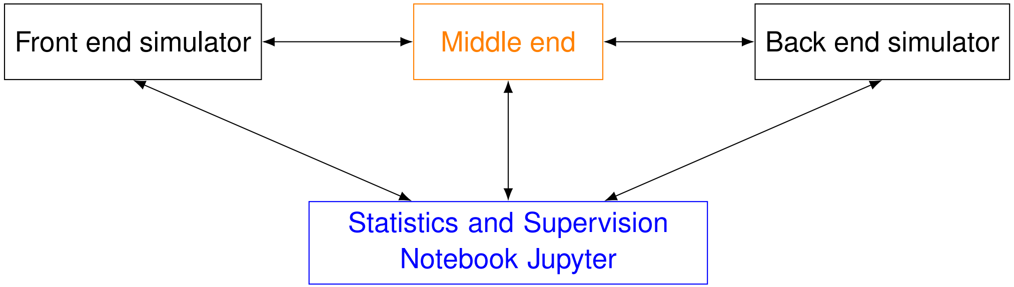
# The case study

A sufficiently realistic e-commerce system



Clients and stocks can vary in number, act at various rate, be faulty...

## The case study code



- main work is on the *middle end*
- analysis and validation by the *notebook*