Migration of pyx4 web application from monolithic to microservices architecture using Ruby on Rails

TABLE OF CONTENTS

01 Objectives

02 Pyx4 - Process Module

03 Definitions

04 Methodology

05 Process & results

06 Lessons & perspectives

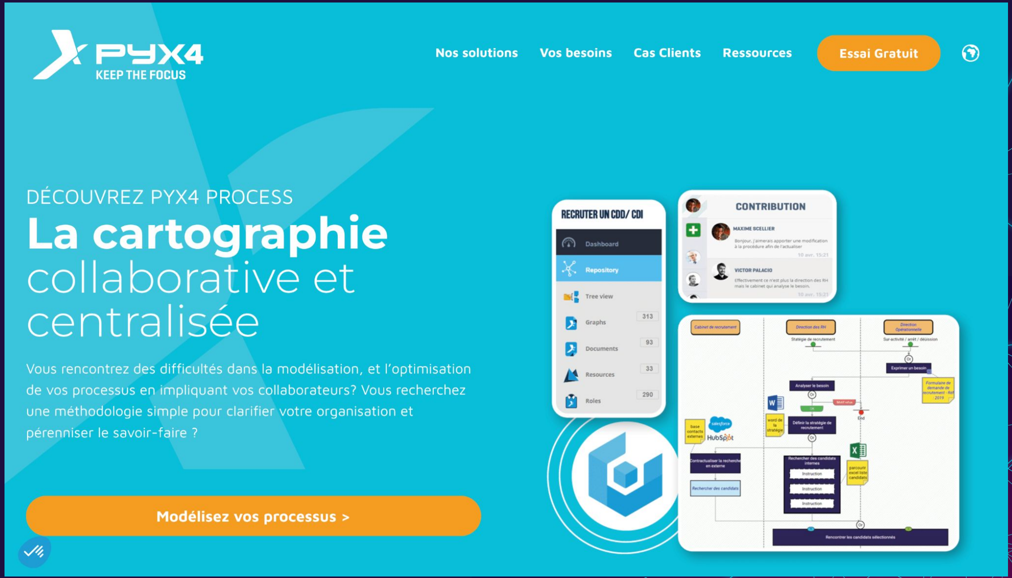
01 Objective

Our goal

Our main goal from this project is the migration of the Pyx4 application from monolithic to microservices architecture

02 Pyx4 – Process module

PYX4 – Process module



DÉCOUVREZ PYX4 PROCESS

La cartographie collaborative et centralisée

Vous rencontrez des difficultés dans la modélisation, et l'optimisation de vos processus en impliquant vos collaborateurs? Vous recherchez une méthodologie simple pour clarifier votre organisation et pérenniser le savoir-faire ?

Modélisez vos processus >

MAXIME SCELLIER
Bonjour, j'aimerais apporter une modification à la procédure afin de l'actualiser
10 avr. 15:21

VICTOR PALACIO
Effectivement ce n'est plus la direction des RH mais le cabinet qui analyse le besoin.
10 avr. 15:21

Pyx4 - process module

Business Side

Technological Side

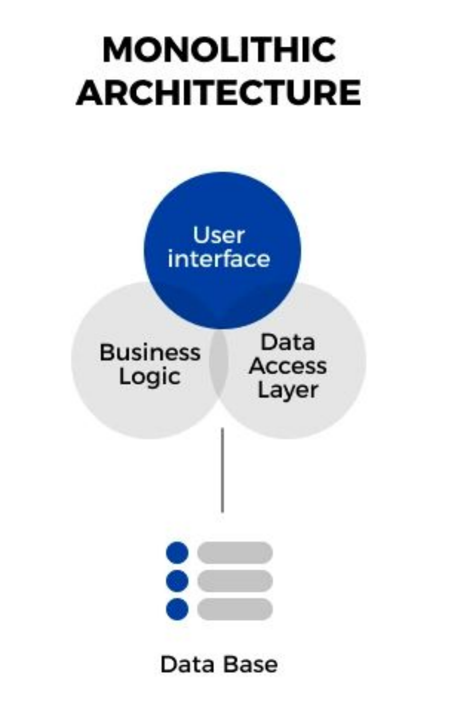
03 Definitions

Monolithic applications

Monolithic architecture is the conventional method of software development.

It is an approach where an entire application is built as a

- **single**, **self-contained unit**.
- components, modules, and functionalities are **tightly coupled and interdependent**.
- All components run within a **single process or instance**.

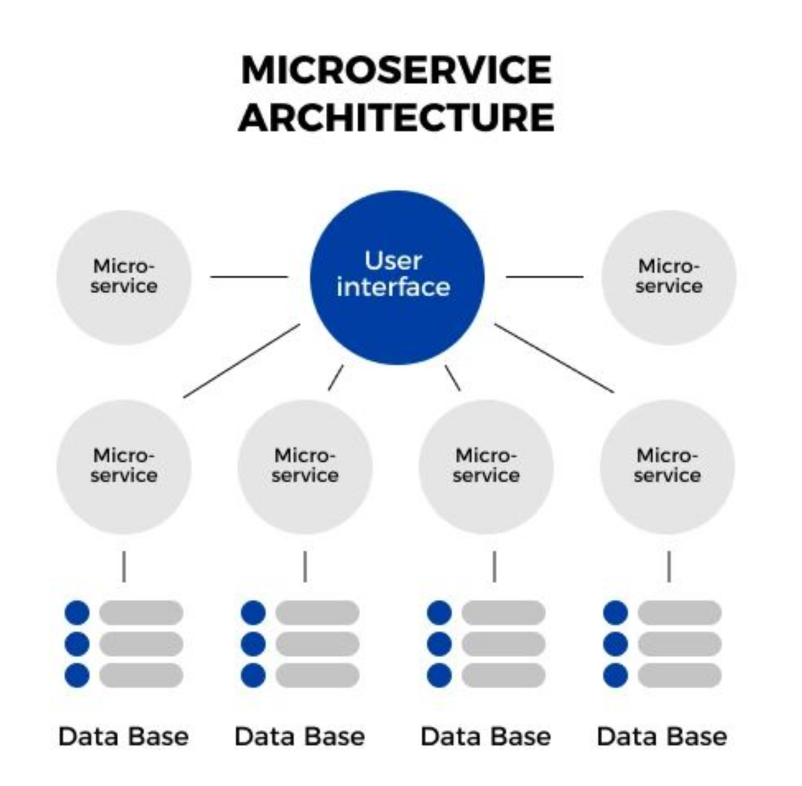


Microservices Architecture

Microservices architecture is an approach where an application is divided into a **collection of small, loosely coupled**, and **independently deployable services**.

Each service:

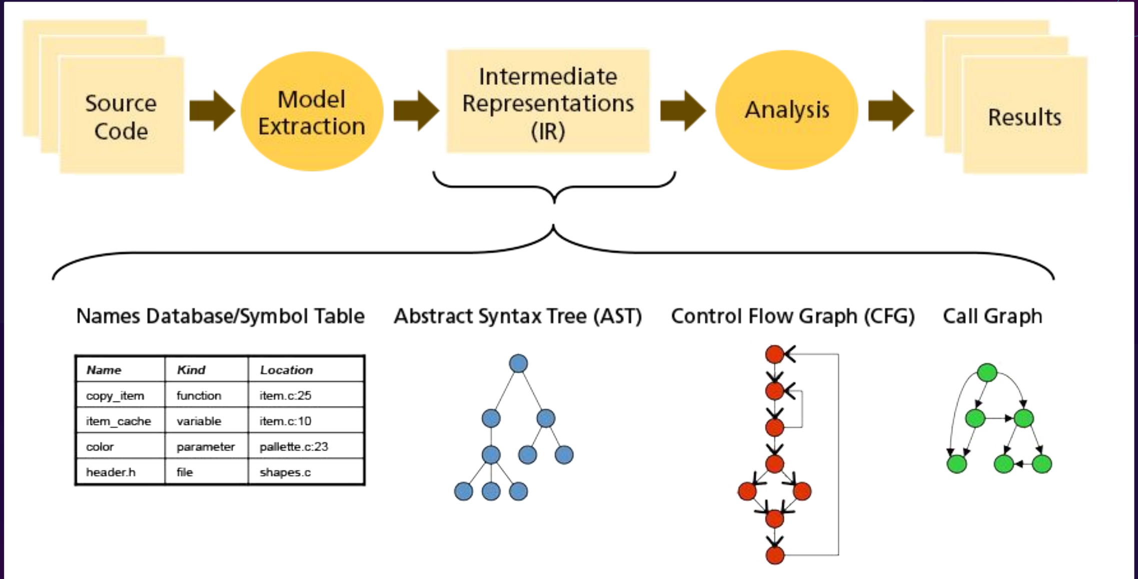
- represents a specific **business capability**.
- runs as a **separate process**
- communicate with other services **through lightweight protocols**.



Monolithic Vs Microservices

	Monolithic	Microservices
Scalability	Requires scaling the entire application	Each service can be independently scaled
Reliability and fault tolerance	A failure in one component can potentially bring down the entire application	A failure in one service does not bring down the entire application
Technology and flexibility	Typically built using a specific technology stack.	Free to choose different technologies based on the service reqiurements.

Static analysis


Source Code
Model Extraction
Intermediate Representations (IR)
Analysis
Results
Names Database/Symbol Table
Abstract Syntax Tree (AST)
Control Flow Graph (CFG)
Call Graph

Name	Kind	Location
copy_item	function	item.c:25
item_cache	variable	item.c:10
color	parameter	pallette.c:23
header.h	file	shapes.c

Ref verysoft-technology

Generation of representative model for our source code → Algorithms for calculation & extraction of properties from the generated models

04 Methodology

Incremental migration

Less risky

Quick passage to production

Validation & non-regression tests

Preserving the visual appearance of the application

Consideration of the maintainability aspect of the migrated application

Database migration is not considered

05 Process & results

A

Identification of microservices

2.1 Documentation

Reading and understanding the structure and basics of Ruby code

Documentation and testing the creation of MS using Ruby on Rails

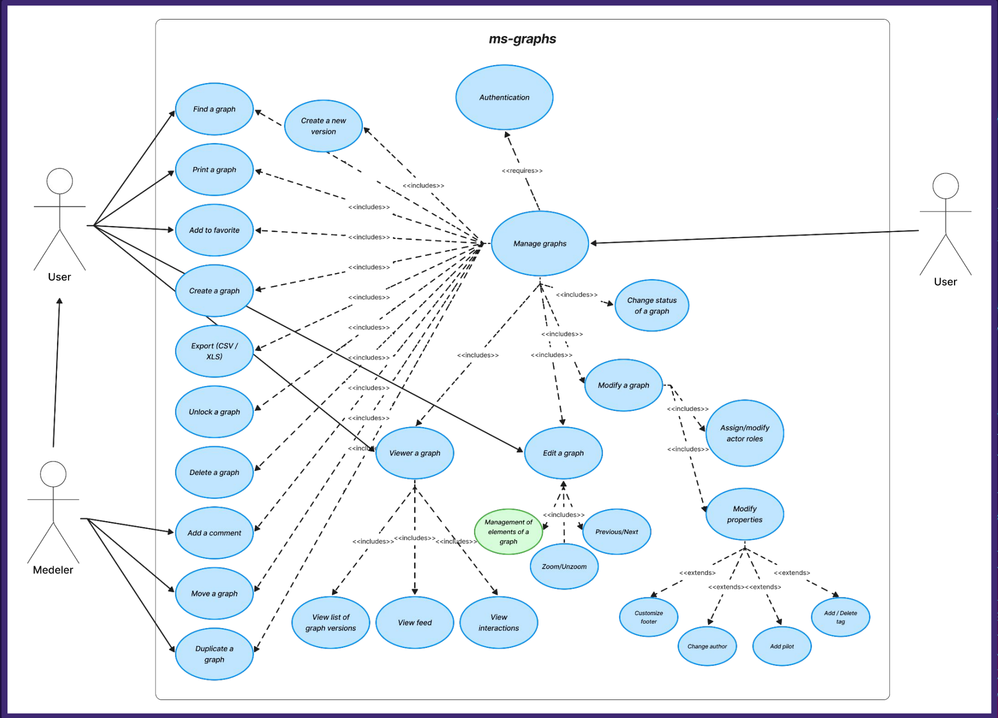
Documentation about static analysis

2.2 Application functioning

Execution and analysis of the features of the application

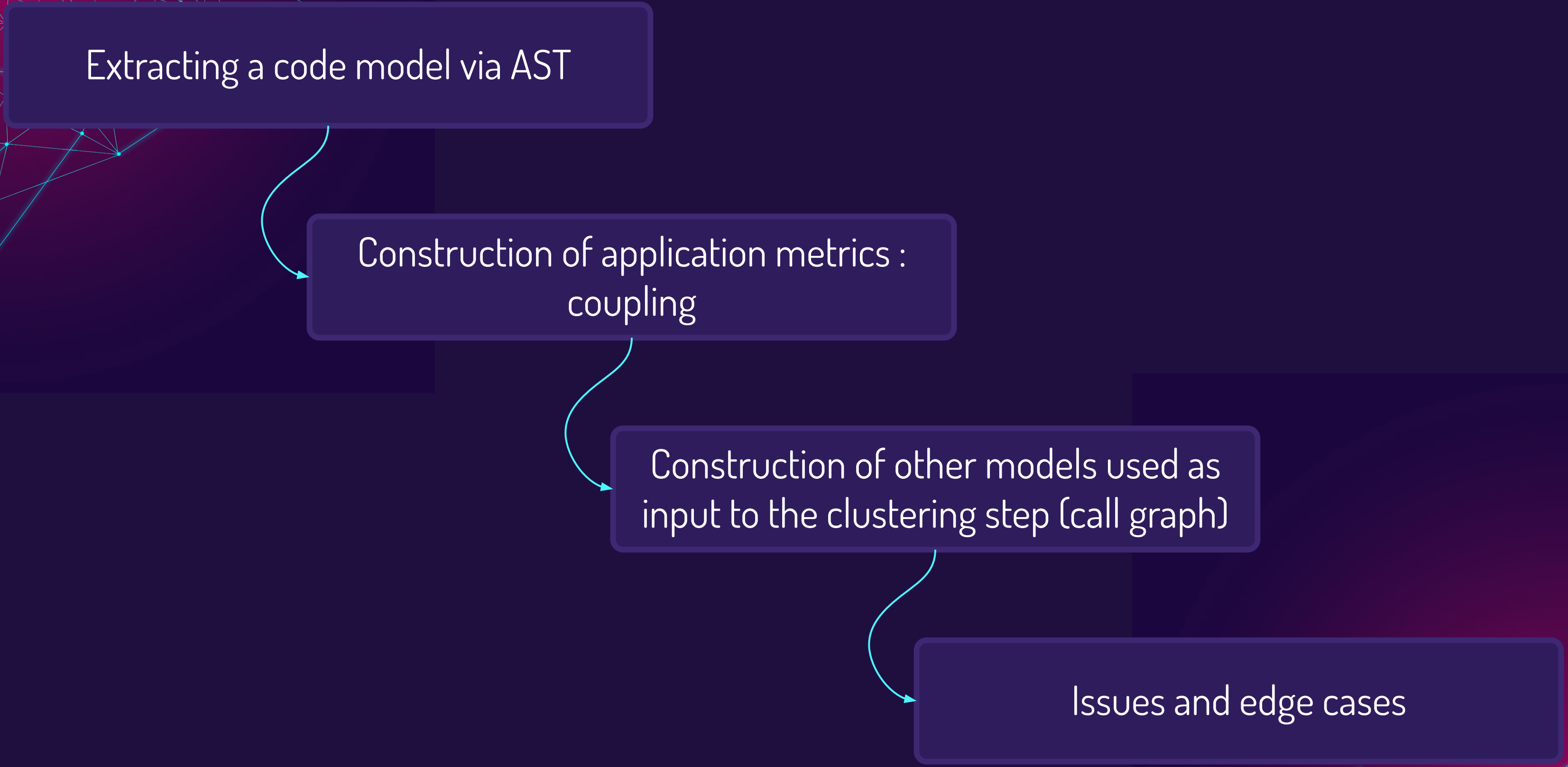
Meeting with application experts

As a result we could extract the application useCase diagrams



Static Analysis on the source code

To understand more the legacy code of the application and its internal behavior, we performed a static analysis on the source code.



Extracting a code model via AST

Construction of application metrics : coupling

Construction of other models used as input to the clustering step (call graph)

Issues and edge cases

Issues and edge cases

1. Nature of Ruby that is a dynamically typed language

2. Construction of the call matrix

3. Possible types of attributes

4. Possible types of method parameters

Static analysis by parsing our Abstract Syntax Tree.

1

Generation of vizual representation (for example callgraphs).

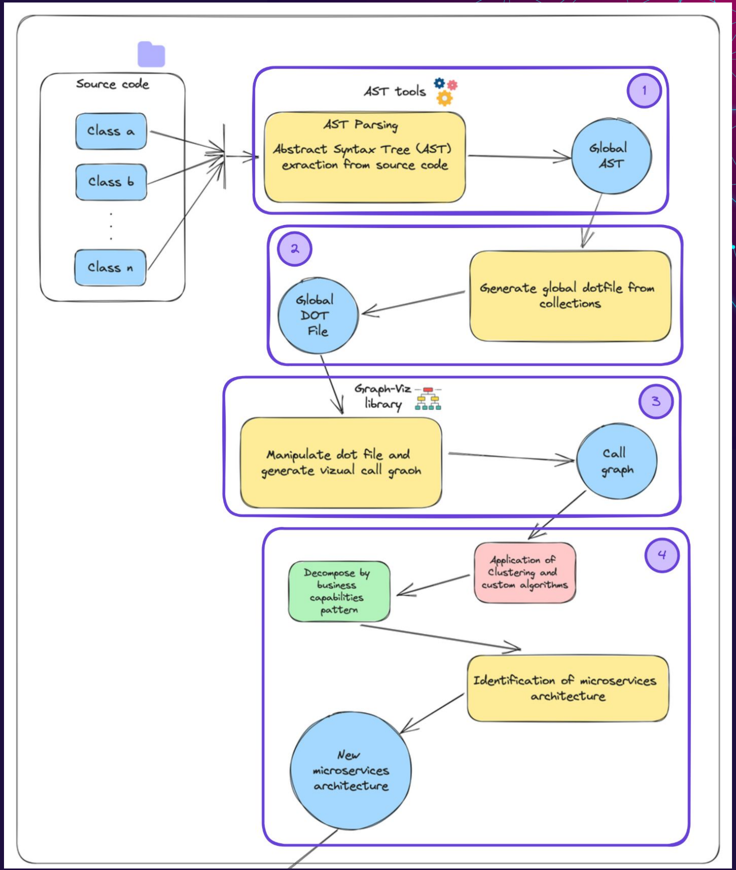
2

Semi-automatic and interactive clustering based on the results of static and manual analysis.

3

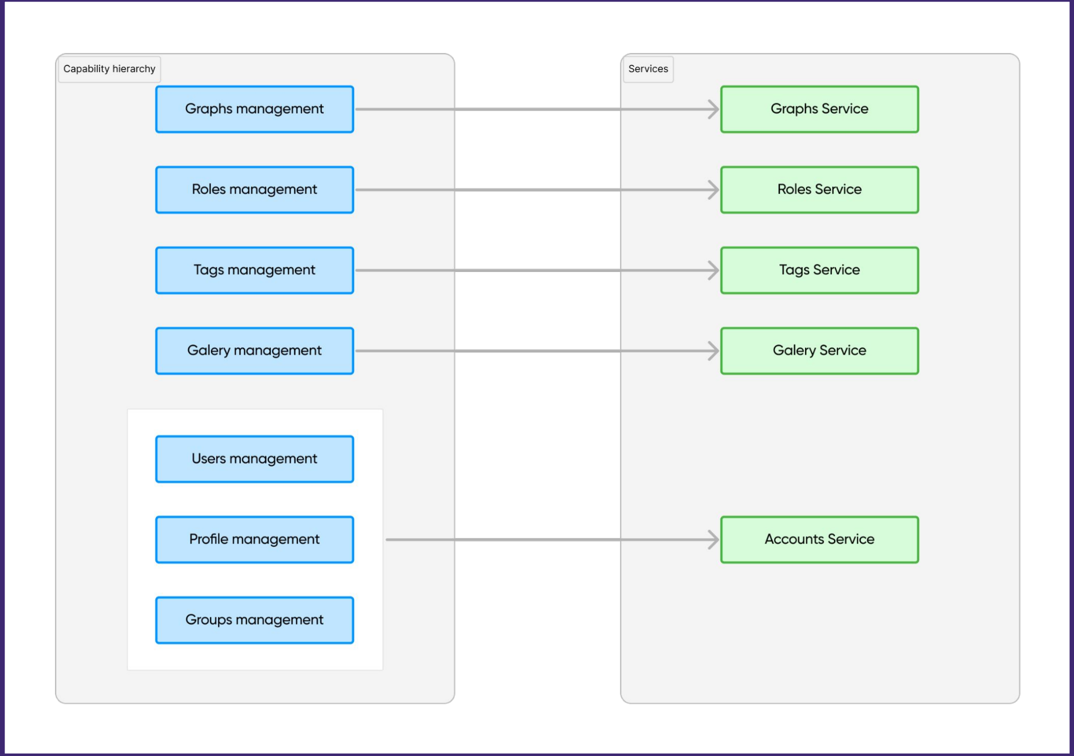
Combinaison with the Decompose by business capabilities pattern.

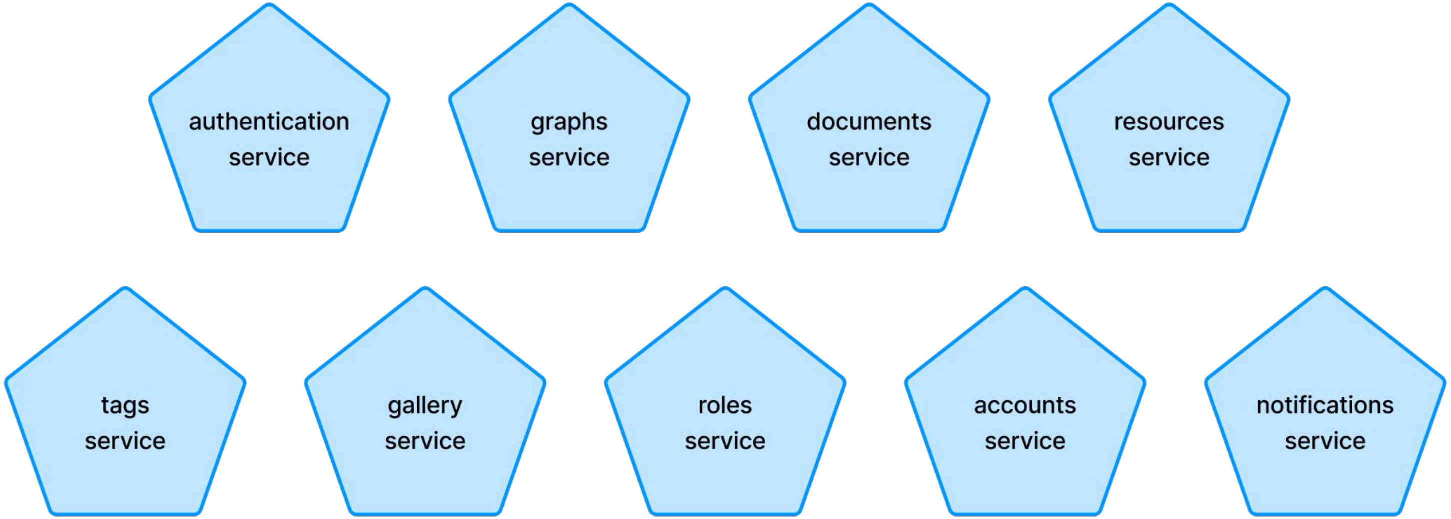
4



Source code
Class a
Class b
Class n
AST tools
1
AST Parsing
Abstract Syntax Tree (AST) exraction from source code
Global AST
2
Generate global dotfile from collections
Global DOT File
Graph-Viz library
3
Manipulate dot file and generate vizual call graoh
Call graph
4
Application of Clustering and custom algorithms
Decompose by business capabilities pattern
Identification of microservices architecture
New microservices architecture

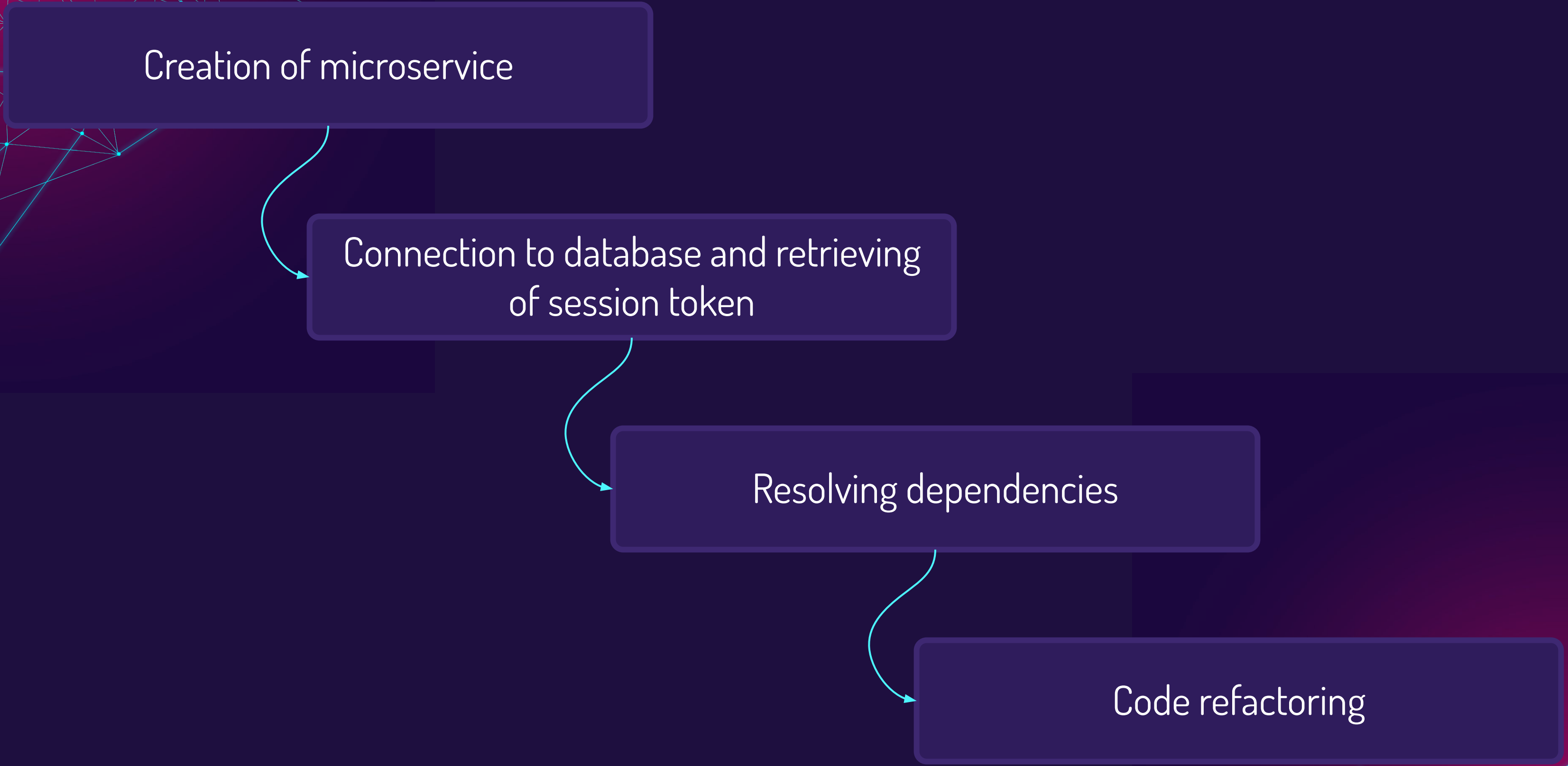
Combine results with manual analysis “Decompose by business capabilities pattern”




authentication service
graphs service
documents service
resources service
tags service
gallery service
roles service
accounts service
notifications service

B

Materialization & incremental migration



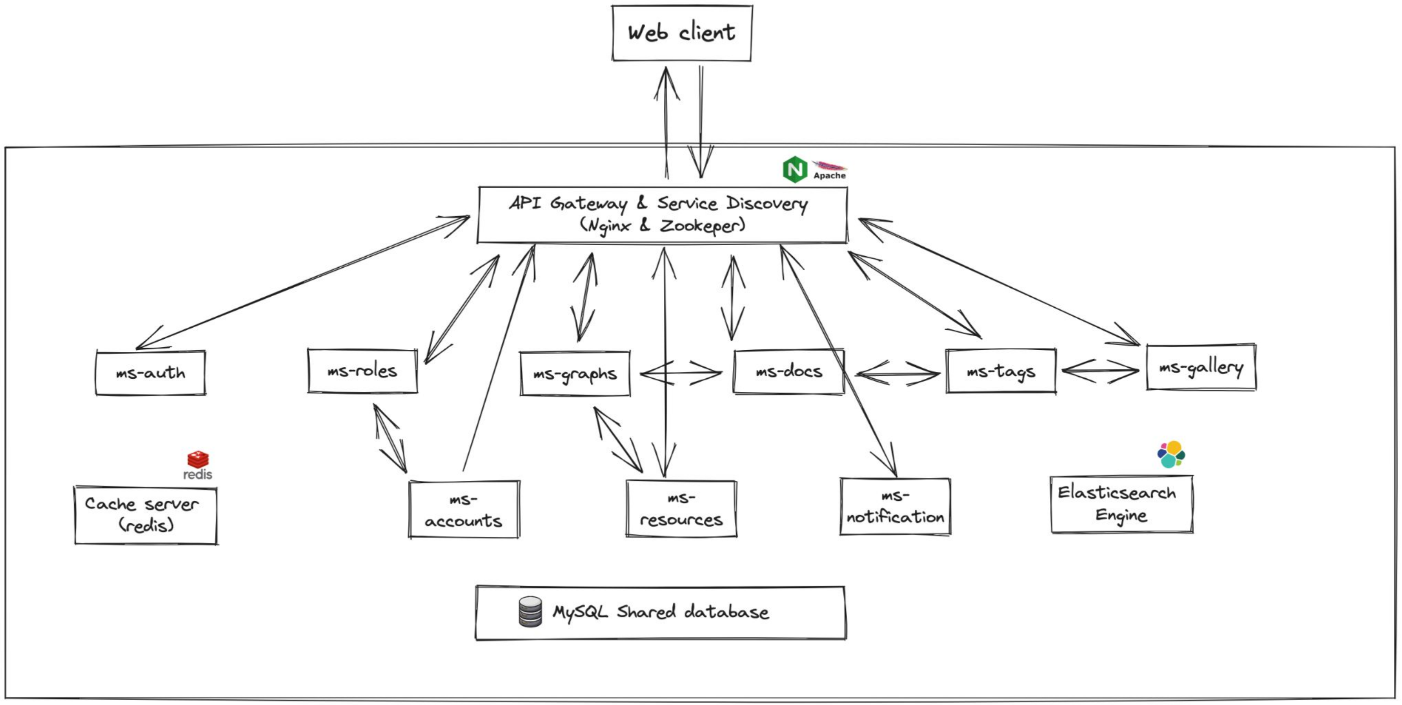
Creation of microservice

Connection to database and retrieving of session token

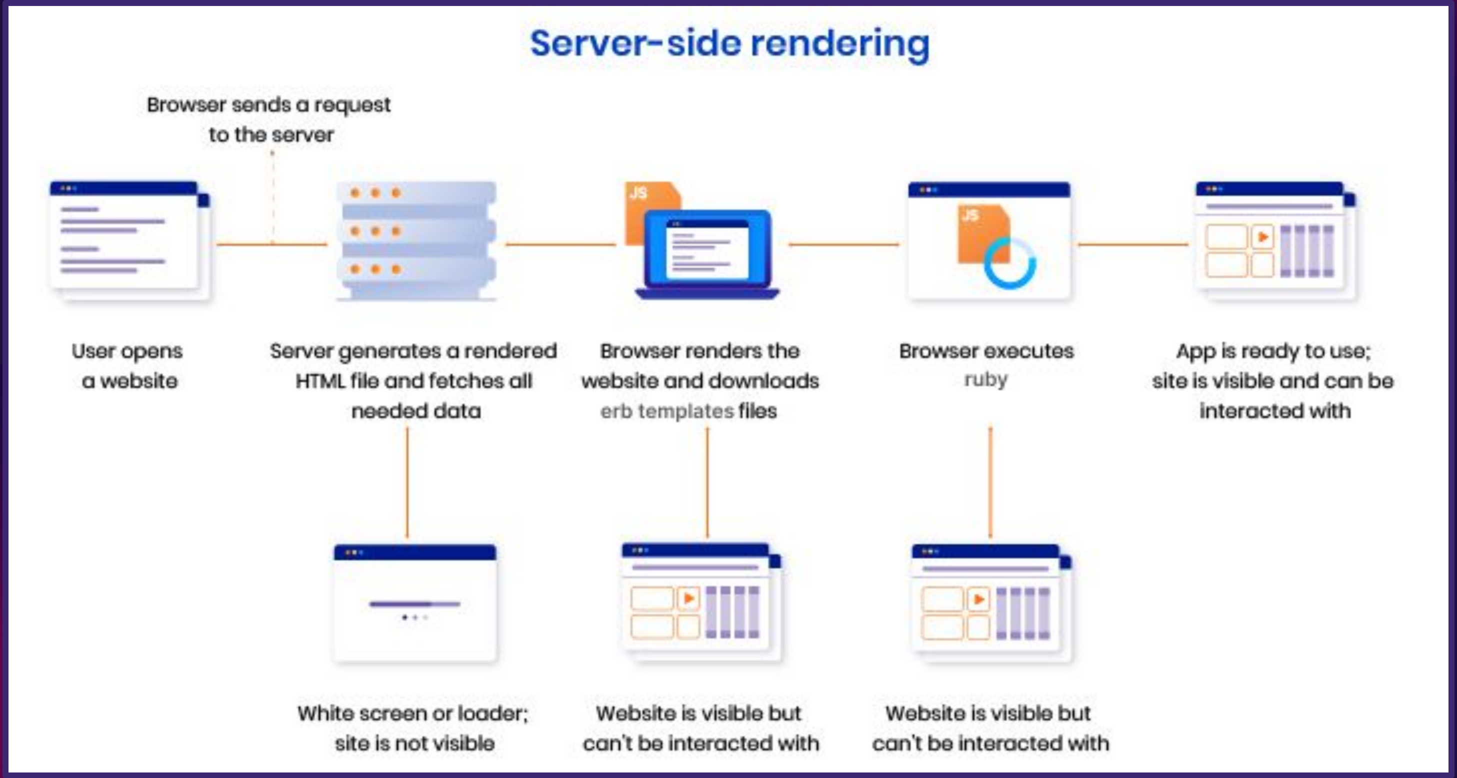
Resolving dependencies

Code refactoring

System global architecture



Isolation of Front-end



Server-side rendering

Browser sends a request to the server

User opens a website

Server generates a rendered HTML file and fetches all needed data

Browser renders the website and downloads erb templates files

Browser executes ruby

App is ready to use; site is visible and can be interacted with

White screen or loader; site is not visible

Website is visible but can't be interacted with

Website is visible but can't be interacted with

DEMO

06

Lessons & Perspectives

Lessons

1. Difficulty of the static analysis on dynamically typed languages

2. Difficulty of transformation front-back relationships

3. Advantages of incremental migration

4. Quality of the leagacy code which complicates understanding

Perspectives

1. Incremental migration of all code

2. Advantages/disadvantages of combining static analysis with dynamic analysis

3. Possibility of technological migration of certain MS

4. Study of the migration of the Database

5. Possibility of coupling the migration process with the evolution of the application

Thank You