PowerEnJoy

Patrizia Porati
Tommaso Sardelli

Politecnico di Milano

February 28, 2017

Introduction

- Highlight most significant features
- Show points of strength in our project
- Emphasize the connection between the four project documents

Most important aspects

1. Difference between application and car systems
2. Architecture and microservices
3. Testing with microservices
4. No constraints on programming language

Analysis

- Definition of **Goals**, based on client requests.

Analysis

- Definition of **Goals**, based on client requests.
- Definition of **Requirements** to fulfill goals.

Analysis

- Definition of **Goals**, based on client requests.
- Definition of **Requirements** to fulfill goals.
- Noticed something interesting

Analysis

- Definition of **Goals**, based on client requests.
- Definition of **Requirements** to fulfill goals.
- Noticed something interesting
- Requirements highlighted **two loosely related components:**
  - User application
  - Cars

Difference App/Car

- Common back end application for apps and cars, **but:**

Difference App/Car

- Common back end application for apps and cars, **but:**
- **Some services required only by one** of the two components (registration, payment, etc.)

Difference App/Car

- Common back end application for apps and cars, **but:**
- **Some services required only by one** of the two components (registration, payment, etc.)
- Some **common services** (Position, Safe Areas)

Difference App/Car

- Common back end application for apps and cars, **but:**
- **Some services required only by one** of the two components (registration, payment, etc.)
- Some **common services** (Position, Safe Areas)
- Presence of **points of interaction** (Reservations, Rides, etc.)

Architecture

- Modularity

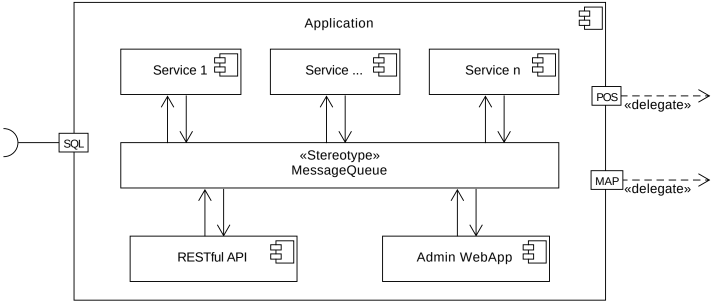
Architecture

- Modularity
- Atomicity

Architecture

- Modularity
- Atomicity
- Interaction between components

The Microservices Idea



Pros and Cons

Pros

Pros and Cons

Pros

- Components can be developed separately

Pros and Cons

Pros

- Components can be developed separately
- Shared components to avoid repetition

Pros and Cons

Pros

- Components can be developed separately
- Shared components to avoid repetition
- Interaction with external system (APIs) only when needed

Pros and Cons

Pros

- Components can be developed separately
- Shared components to avoid repetition
- Interaction with external system (APIs) only when needed

Cons

Pros and Cons

Pros

- Components can be developed separately
- Shared components to avoid repetition
- Interaction with external system (APIs) only when needed

Cons

- Harder to develop

Pros and Cons

Pros

- Components can be developed separately
- Shared components to avoid repetition
- Interaction with external system (APIs) only when needed

Cons

- Harder to develop
- Harder to deploy

Pros and Cons

Pros

- Components can be developed separately
- Shared components to avoid repetition
- Interaction with external system (APIs) only when needed

Cons

- Harder to develop
- Harder to deploy

Bonus

Pros and Cons

Pros

- Components can be developed separately
- Shared components to avoid repetition
- Interaction with external system (APIs) only when needed

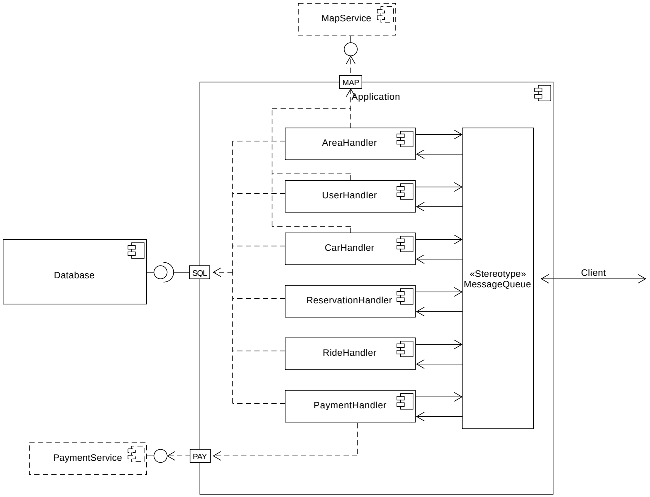
Cons

- Harder to develop
- Harder to deploy

Bonus

- More flexibility for future/unforeseen requirements

Components Diagram

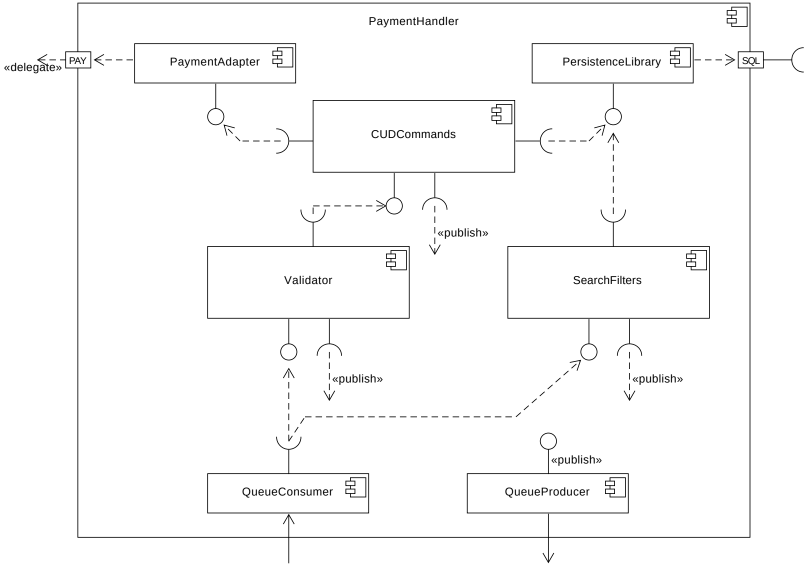


Testing

- **Microservices**
  - components operate indipendently
  - flexible tests order

Testing

- **Microservices**
  - components operate indipendently
  - flexible tests order
- **Bottom-up strategy:**
  testing can start even if other components aren't completely developed
  1. unit testing of subcomponents
  2. integration testing between subcomponents (and external services)
  3. integration testing between components
  4. system testing


PaymentHandler
PaymentAdapter
PersistenceLibrary
«delegate»
PAY
SQL
CUDCommands
«publish»
Validator
SearchFilters
«publish»
«publish»
«publish»
QueueConsumer
QueueProducer

Programming language

- In these documents we omit specific details like
  - programming languages
  - software to use across the system

  to leave these choices to developers

Lines of Code

LoC = AVG * FPs

Lines of Code

$$\textbf{LoC} = \textbf{AVG} * \textbf{FPs}$$

- **AVG** language-dependent factor
- **FPs** total number of Function Points

JavaScript

- In order to calculate the *effort* (person-month) and the *duration* and make the **project plan** we need to choose a representative programming language.
- We choose **JavaScript** as an high level language example.