Model-Based Approach to Consume REST Services in Single Page Applications

Niklas Scholz - Master Thesis Kickoff - 19.06.2017

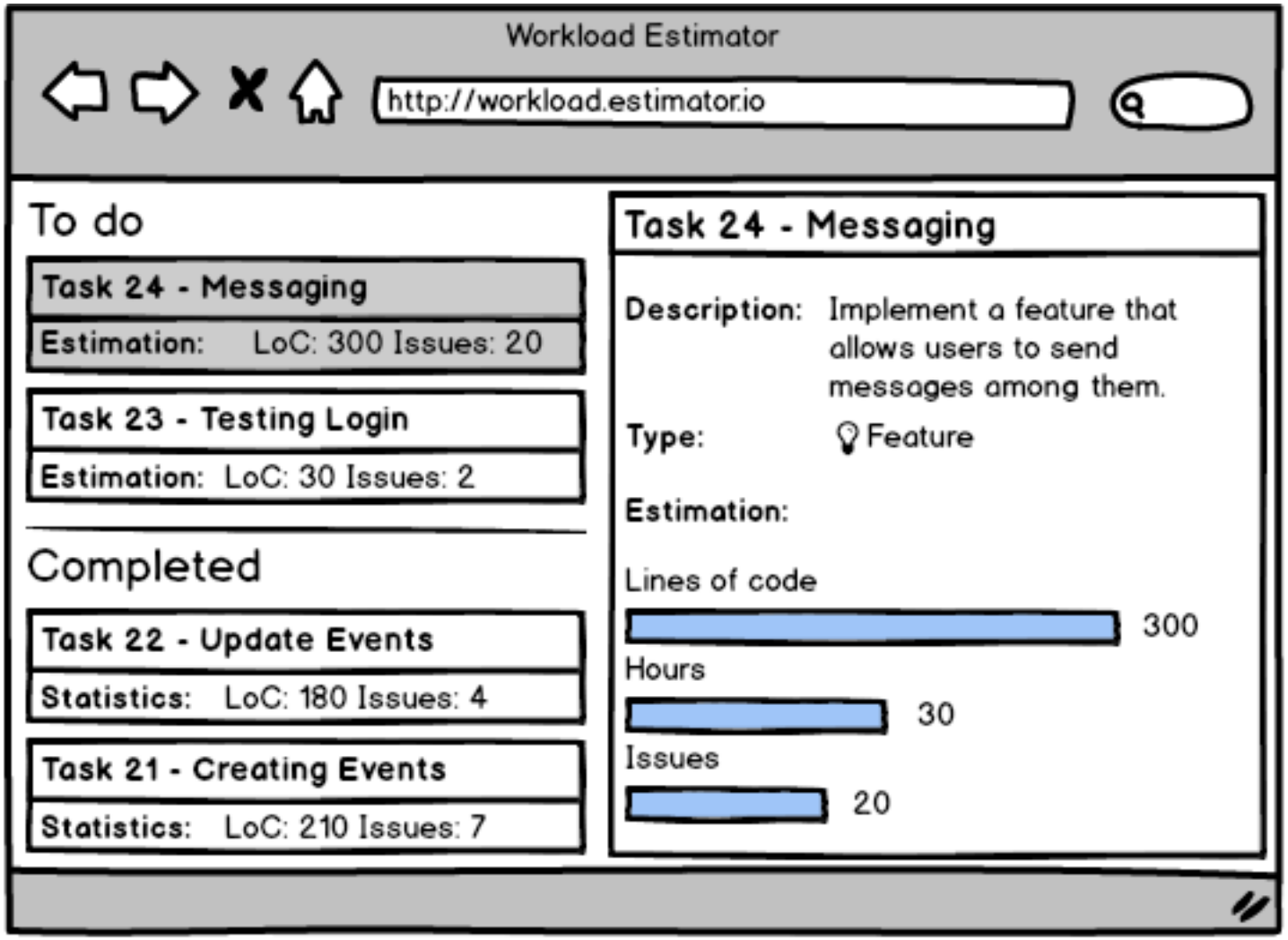
Advisor: Adrian Hernandez-Mendez

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

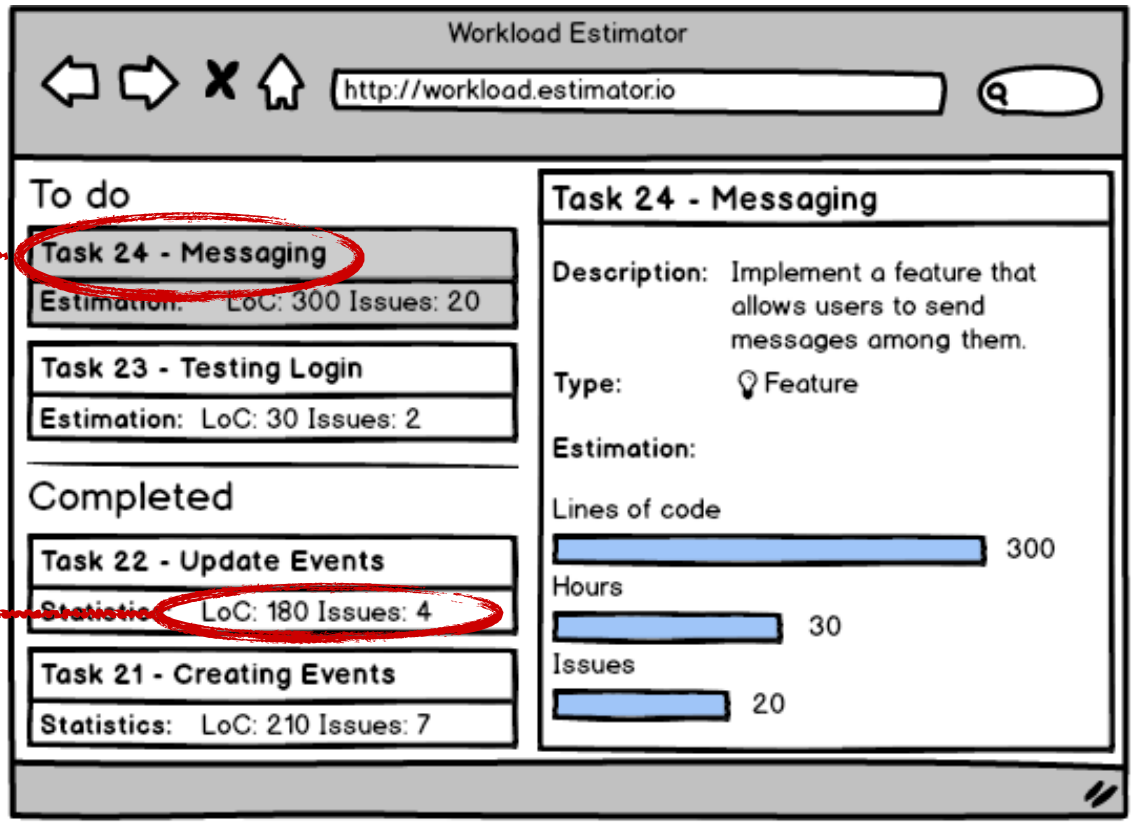
Agenda

1. Motivation and Problem
2. Research Questions
3. Existing Solutions
4. Limitations
5. Our solution
6. Expected artefacts & Timeline

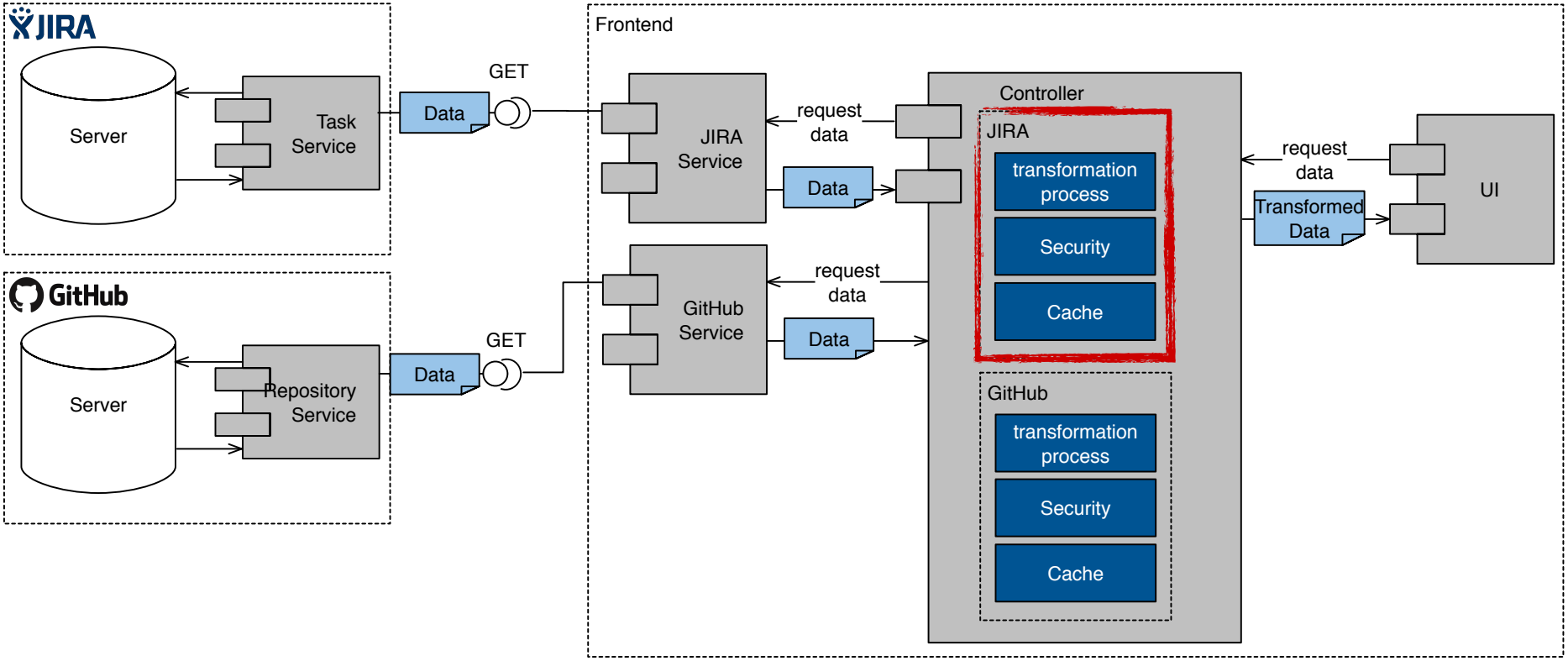
Building an Application in 2017



Where does the data come from?



Architecture of such an application



Problem

For each API manage:

- Transformation process
- Cache
- Security

Leads to overhead in controller and makes it hard to change UI

Research Questions

1. **How to retain a lightweight frontend when scaling up the number of APIs to consume?**
   - How to handle data transformation, cache and security for each API?
2. **How does a model for service consumption of multiple APIs look like?**
   - What information is needed from existing models? (View Model, Data Model, REST API Model…)
3. **How does a technology independent approach look like that can be used with any API?**
   - Can this model be used to generate the code for the consumption of APIs?

Model-Based User Interface Development

- Develop software based on explicit models
- Reference Architecture
- do (semi-) automatic code generation
- High focus on platform independent UI models

Researching Solutions

Approaches dealing with similar problem:

SOA

Architectural style that focusses on structuring and using services

apigee

Managing APIs
Providing:
- Gateway
- Security
- Analytics
- …

WSO2

Design, Create, Publish, Manage APIs

GraphQL

Query Language for Graph APIs

Limitations

- No model provided
- No clear specification
- WSO2 and apigee
  - complex
  - we don’t have control over server

REST API Specification

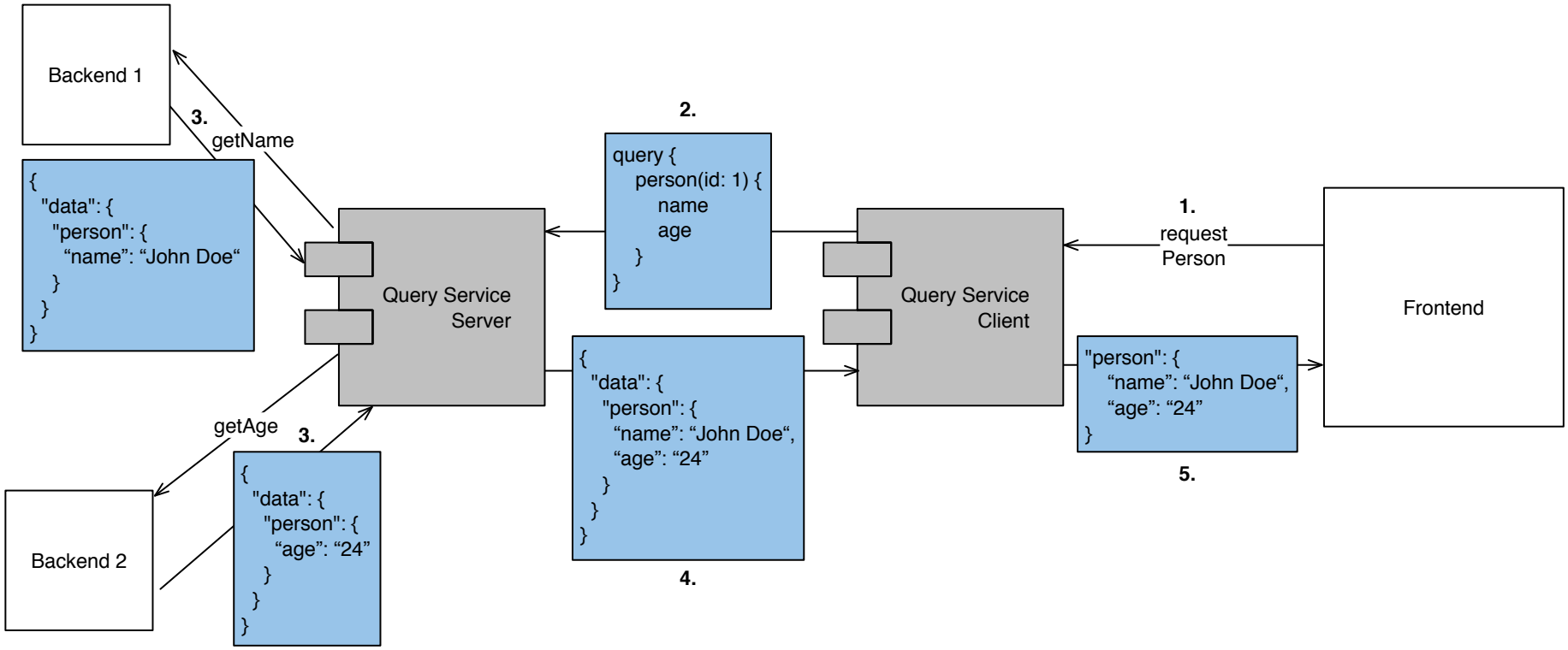
- Describing RESTful APIs for example with:

- Specify
  - accessible resources (in path element)
  - Data Model (in definitions element)
  - Security (authentication)

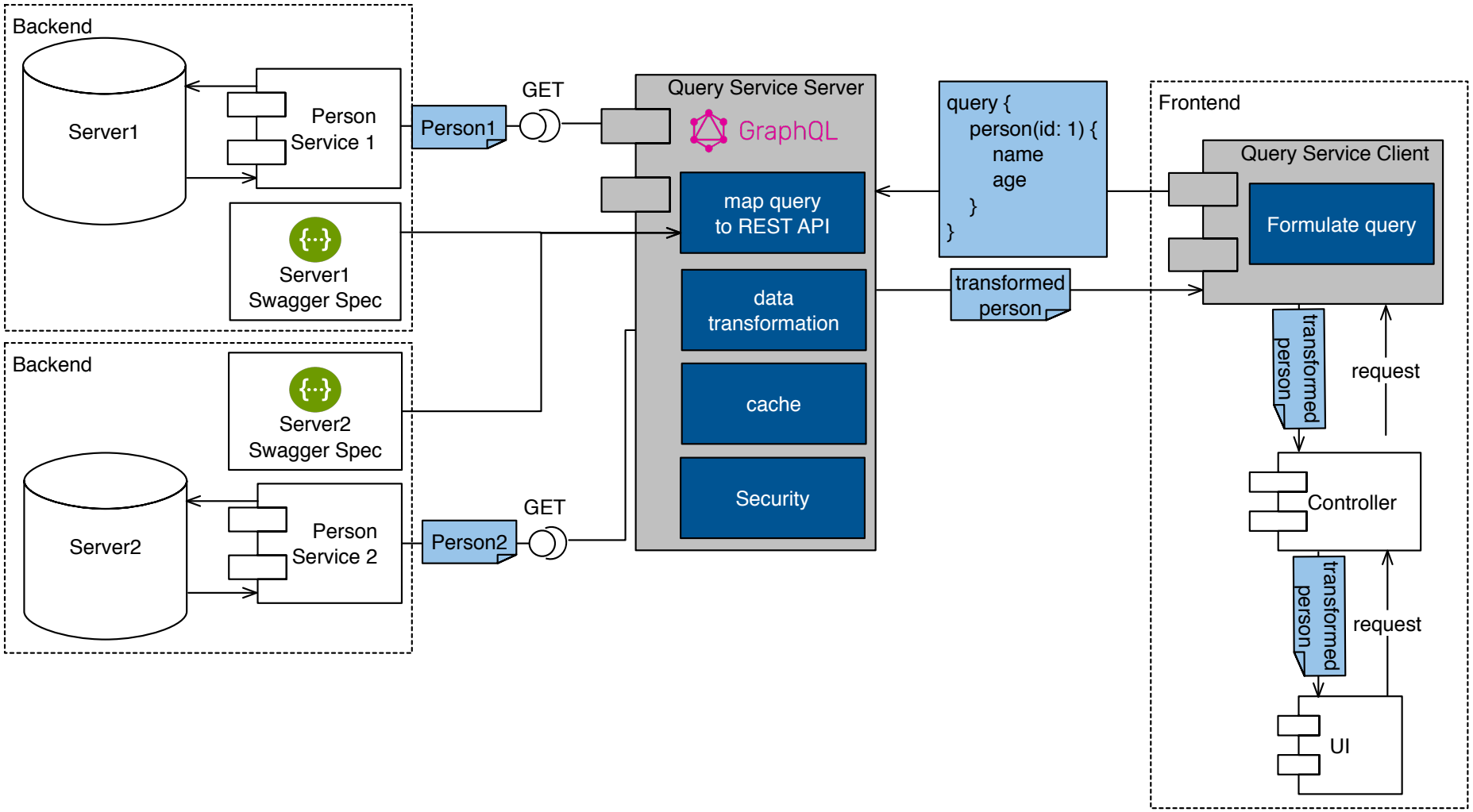
REST API Specification

- Reusing these concepts can help reducing complexity on server
- How to reuse this concept in an architecture?

Solution: Query Service



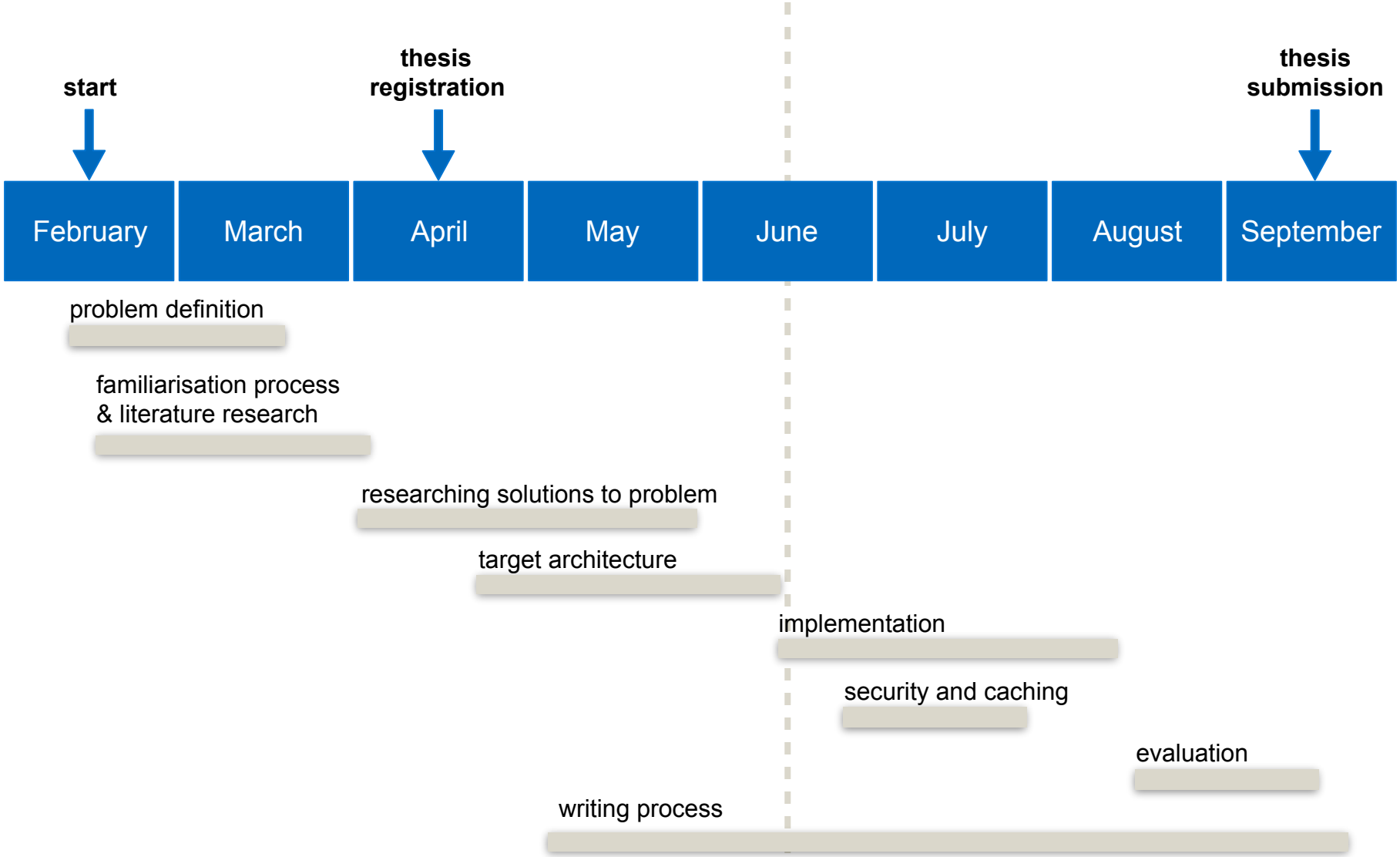
Architecture of Query Service



Advantages of This Approach

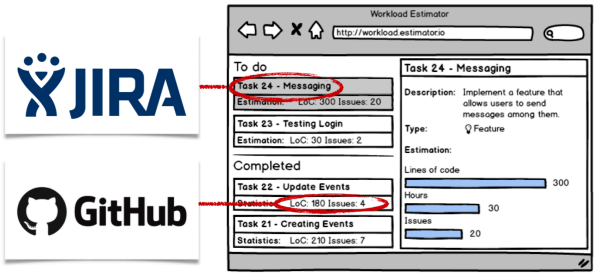
- Keep client simple
  - Important for Model-Based User Interfaces
  - Shifting responsibility away form client but not completely to server
- Provide actual server
  - Deliver together with client part
  - Less complex server compared to creating an actual backend
- Ability to deal with complex queries (thanks to GraphQL)
  - Simplify data transformation process

Timeline

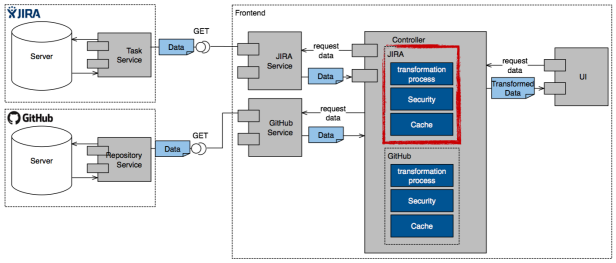


Summing up

Where does the data come from?



Architecture of such an application



Researching Solutions

Approaches dealing with similar problem:

SOA

Architectural style that focusses on structuring and using services

apigee

Managing APIs Providing:
- Gateway
- Security
- Analytics
- …

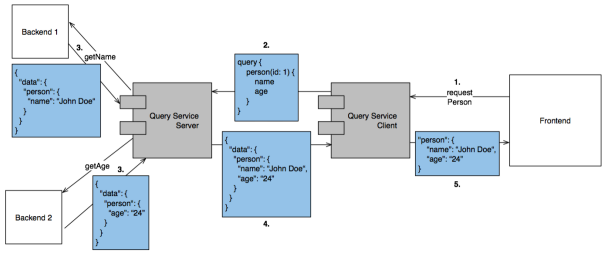
WSO2

Design, Create, Publish, Manage APIs

GraphQL

Query Language for Graph APIs

Solution: Query Service



Advantages of This Approach

- Keep client simple
  - Important for UI Driven Development
  - Shifting responsibility away form client but not completely to server
- Provide actual server
  - Deliver it together with client part
  - Less complex server if you would create an actual backend
- Ability to deal with complex queries (thanks to GraphQL)
  - Simplify data transformation process