

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

Niklas Scholz - Final Presentation Master Thesis - 09.10.2017

Advisor: Adrian Hernandez-Mendez

Chair of Software Engineering for Business Information Systems (sebis)  
Faculty of Informatics  
Technische Universität München  
[www.matthes.in.tum.de](http://www.matthes.in.tum.de)

# Agenda

Motivation and Problem

Research Questions

Background

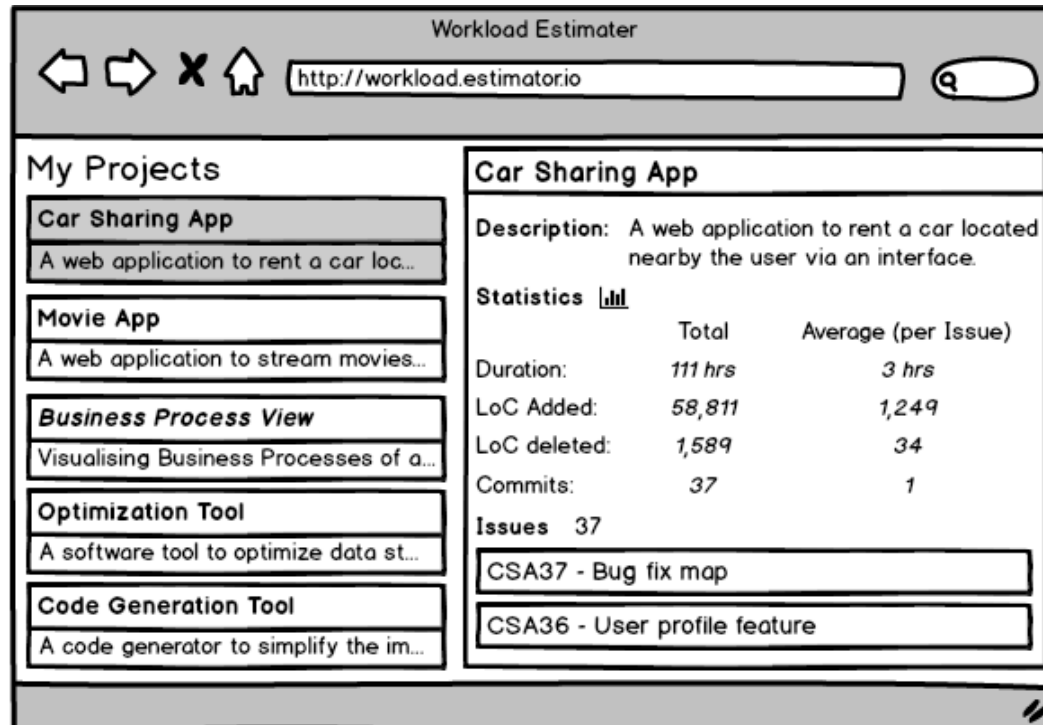
Approach

- Reference Architecture
- Meta-Model
- Code Generation Process

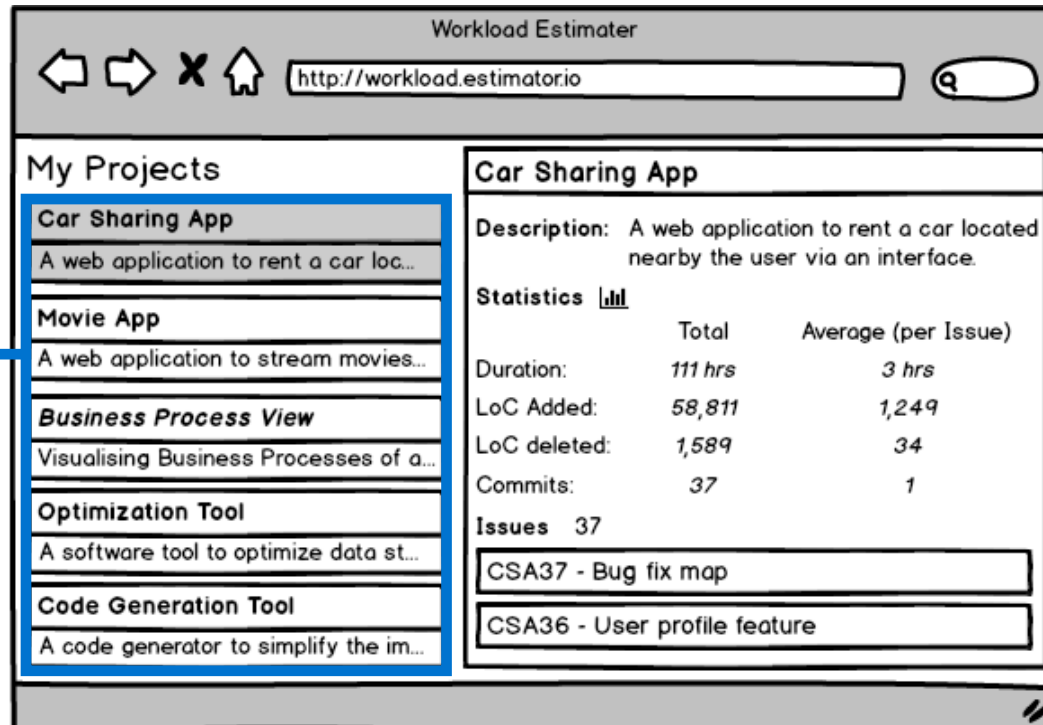
Live Demo

Evaluation





# Building an Application in 2017



Workload Estimator


http://workload.estimator.io

### My Projects

- Car Sharing App**  
A web application to rent a car loc...
- Movie App**  
A web application to stream movies...
- Business Process View**  
Visualising Business Processes of a...
- Optimization Tool**  
A software tool to optimize data st...
- Code Generation Tool**  
A code generator to simplify the im...

### Car Sharing App

**Description:** A web application to rent a car located nearby the user via an interface.

**Statistics** 

	Total	Average (per Issue)
Duration:	111 hrs	3 hrs
LoC Added:	58,811	1,249
LoC deleted:	1,589	34
Commits:	37	1

**Issues** 37

- CSA37 - Bug fix map
- CSA36 - User profile feature



# Building an Application in 2017

The screenshot shows a web browser window titled "Workload Estimator" with the URL "http://workload.estimator.io". The interface is divided into two main sections: "My Projects" on the left and "Car Sharing App" details on the right. The "My Projects" list includes "Car Sharing App", "Movie App", "Business Process View", "Optimization Tool", and "Code Generation Tool". The "Car Sharing App" section displays a description, a statistics table, and a list of issues.

	Total	Average (per Issue)
Duration:	111 hrs	3 hrs
LoC Added:	58,811	1,249
LoC deleted:	1,589	34
Commits:	37	1

Issues: 37

- CSA37 - Bug fix map
- CSA36 - User profile feature

# Building an Application in 2017

Workload Estimator

http://workload.estimator.io

### My Projects

- Car Sharing App**  
A web application to rent a car loc...
- Movie App**  
A web application to stream movies...
- Business Process View**  
Visualising Business Processes of a...
- Optimization Tool**  
A software tool to optimize data st...
- Code Generation Tool**  
A code generator to simplify the im...

### Car Sharing App




Description: A web application to rent a car located nearby the user via an interface.

Statistics

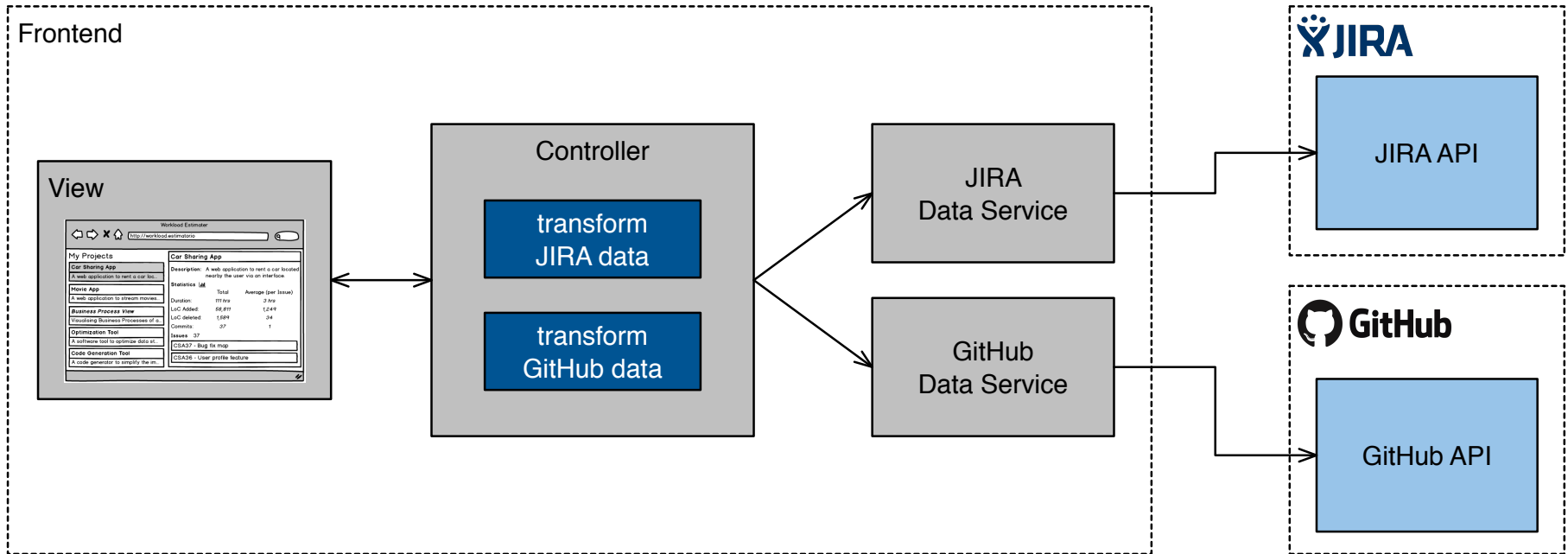
	Total	Average (per Issue)
Duration:	111 hrs	3 hrs
LoC Added:	58,811	1,249
LoC deleted:	1,589	34
Commits:	37	1

Issues 37

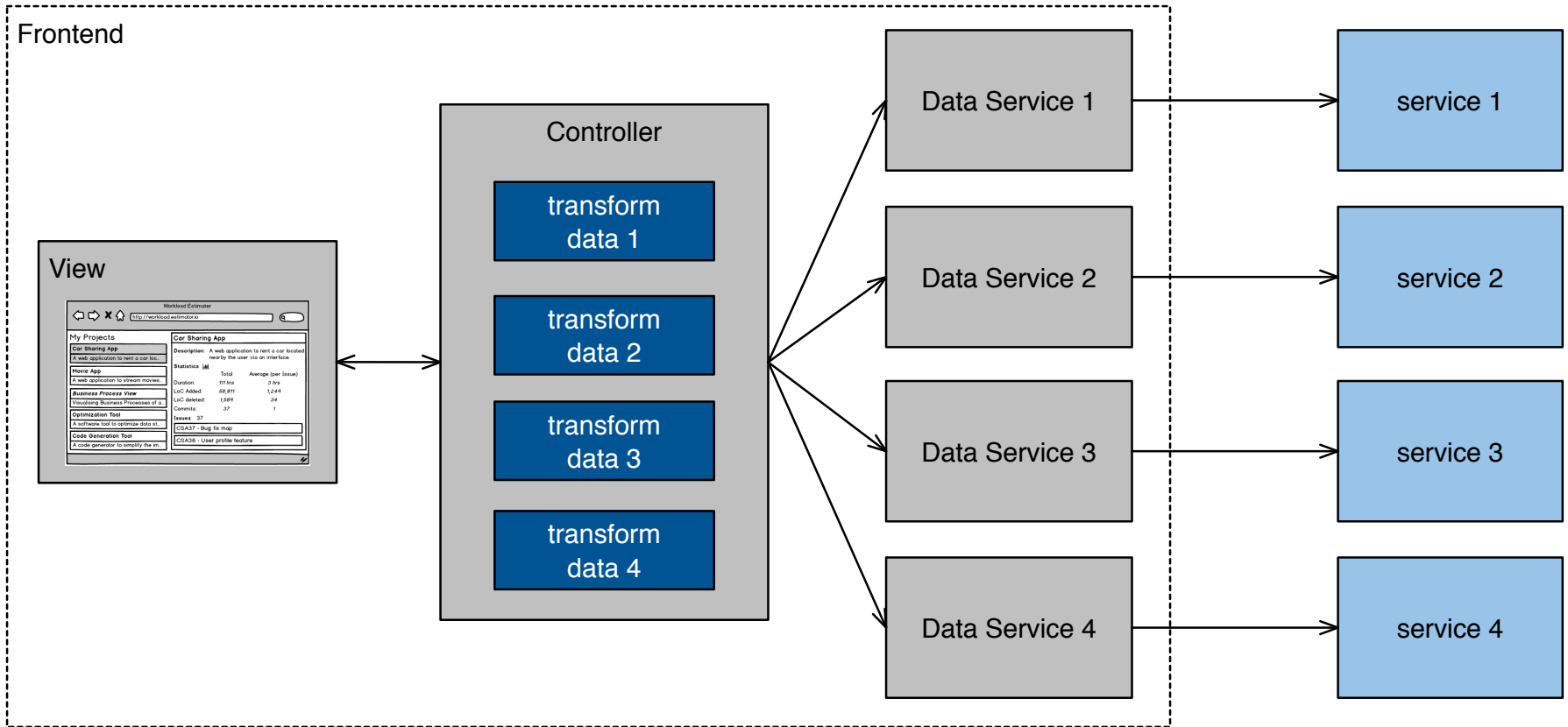
- CSA37 - Bug fix map
- CSA36 - User profile feature



# Consuming APIs in Single Page Applications

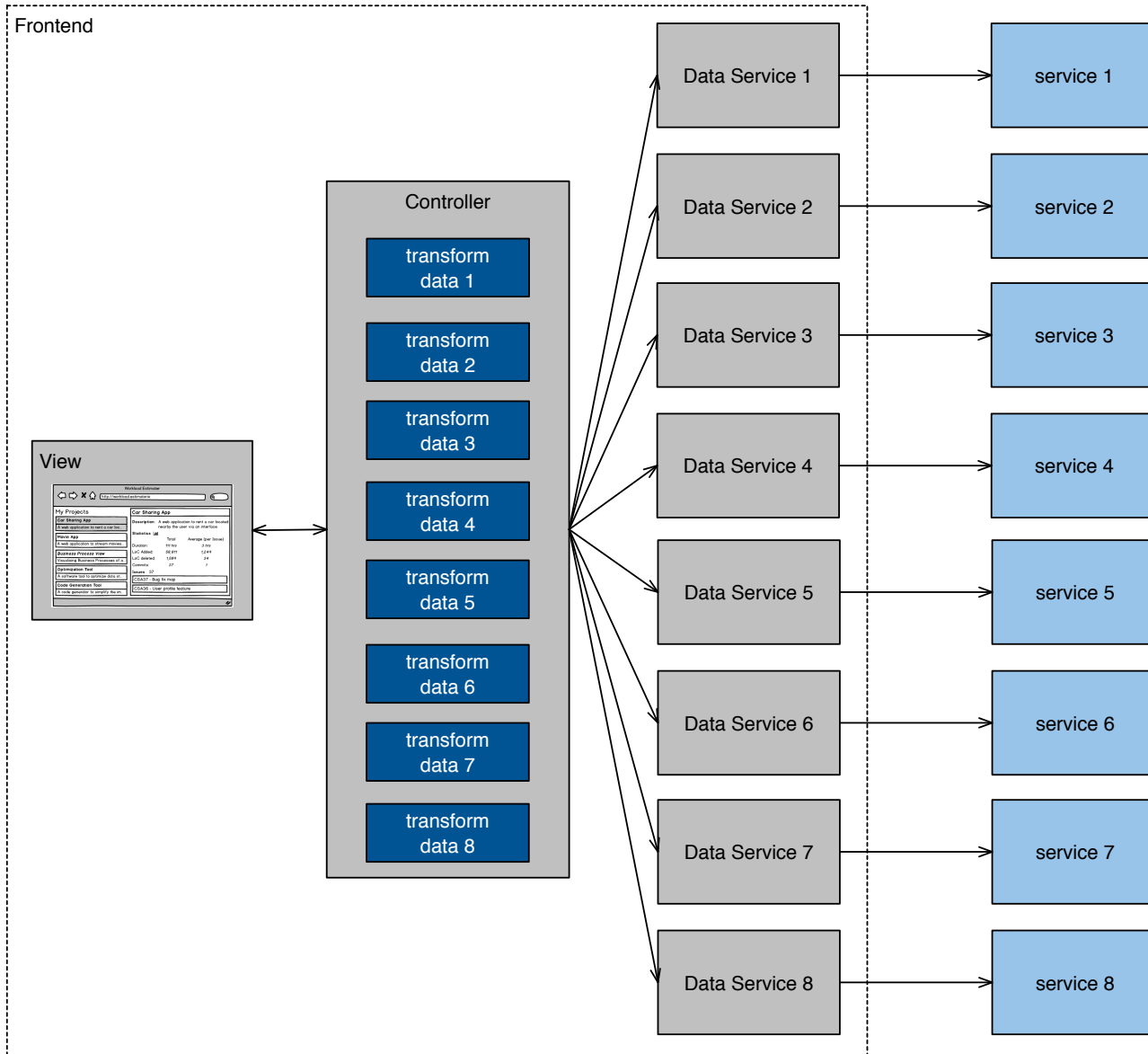


# Scaling up number of APIs to consume





# Scaling up number of APIs to consume



**RQ1**

**What is the state-of-the-art in model-based RESTful API integration?**

**RQ1**

**What is the state-of-the-art in model-based RESTful API integration?**

**RQ2**

**How does a model-based approach for RESTful API integration in single page applications look like?**

**RQ1**

**What is the state-of-the-art in model-based RESTful API integration?**

**RQ2**

**How does a model-based approach for RESTful API integration in single page applications look like?**

**RQ3**

**What are the benefits and limitations of a model-driven approach for RESTful API integration?**

- Lanthaler and Gütl: Hydra [1]
  - RESTful API description language
- Rossi: Model-driven REST API development [2]
  - Constricting model in UML and transforming it to RAML
- Modelling REST with the Eclipse Modelling Framework (EMF)
  - Ed-Douibi et al.: Ready-to-run web APIs out of data models [3]
  - Haupt et al.: Generate REST compliant services [4]
- Bonifacio et al.: NeoIDL [5]
  - domain specific language to specify RESTful APIs

## Web development frameworks



## API Design Guidelines



## API Description



## API Management Tools

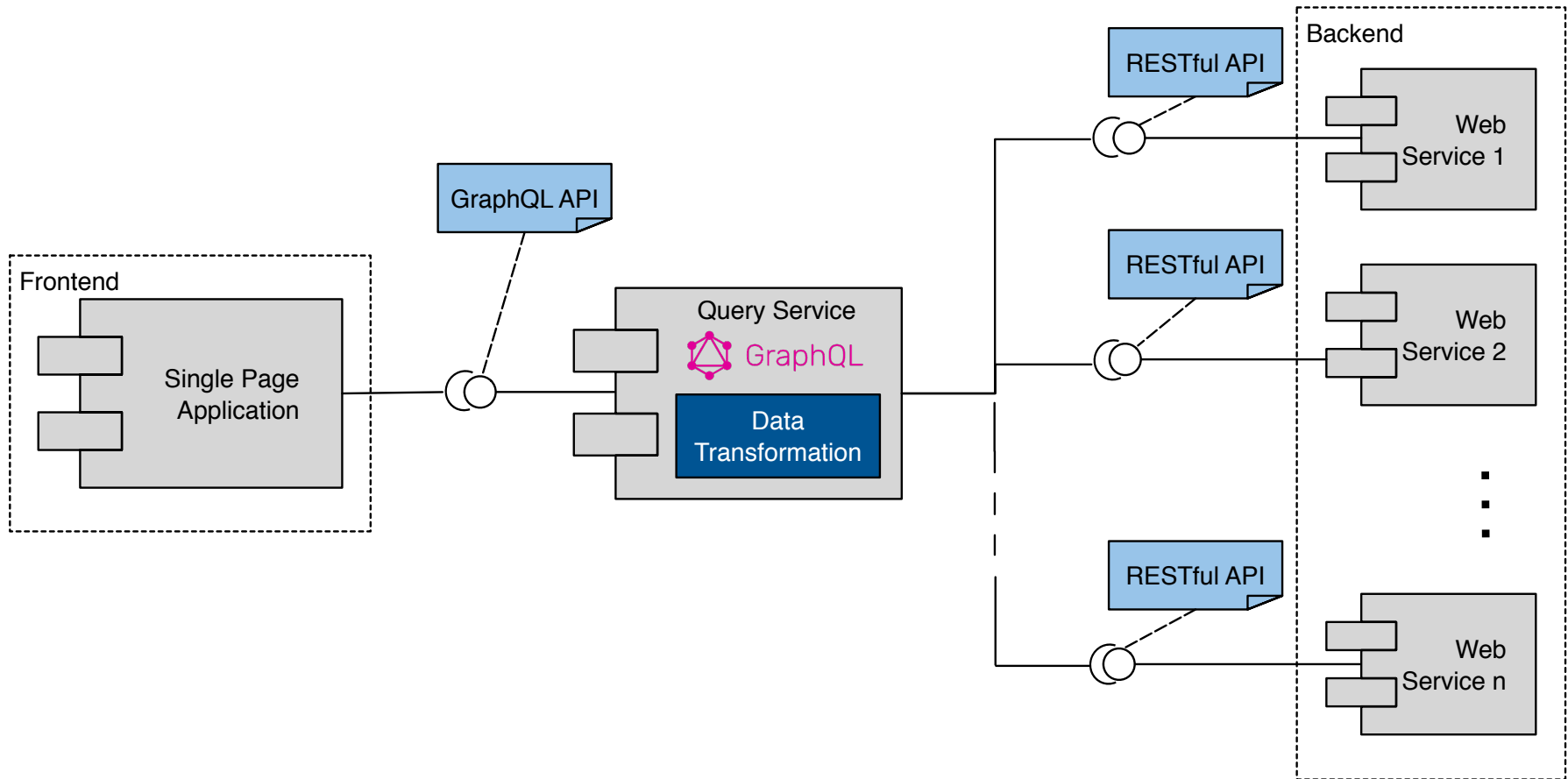




## Reference Architecture

- Architecture to reduce complexity in client
- shift responsibility of RESTful API consumption outside of client

# Reference Architecture

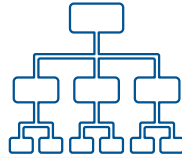






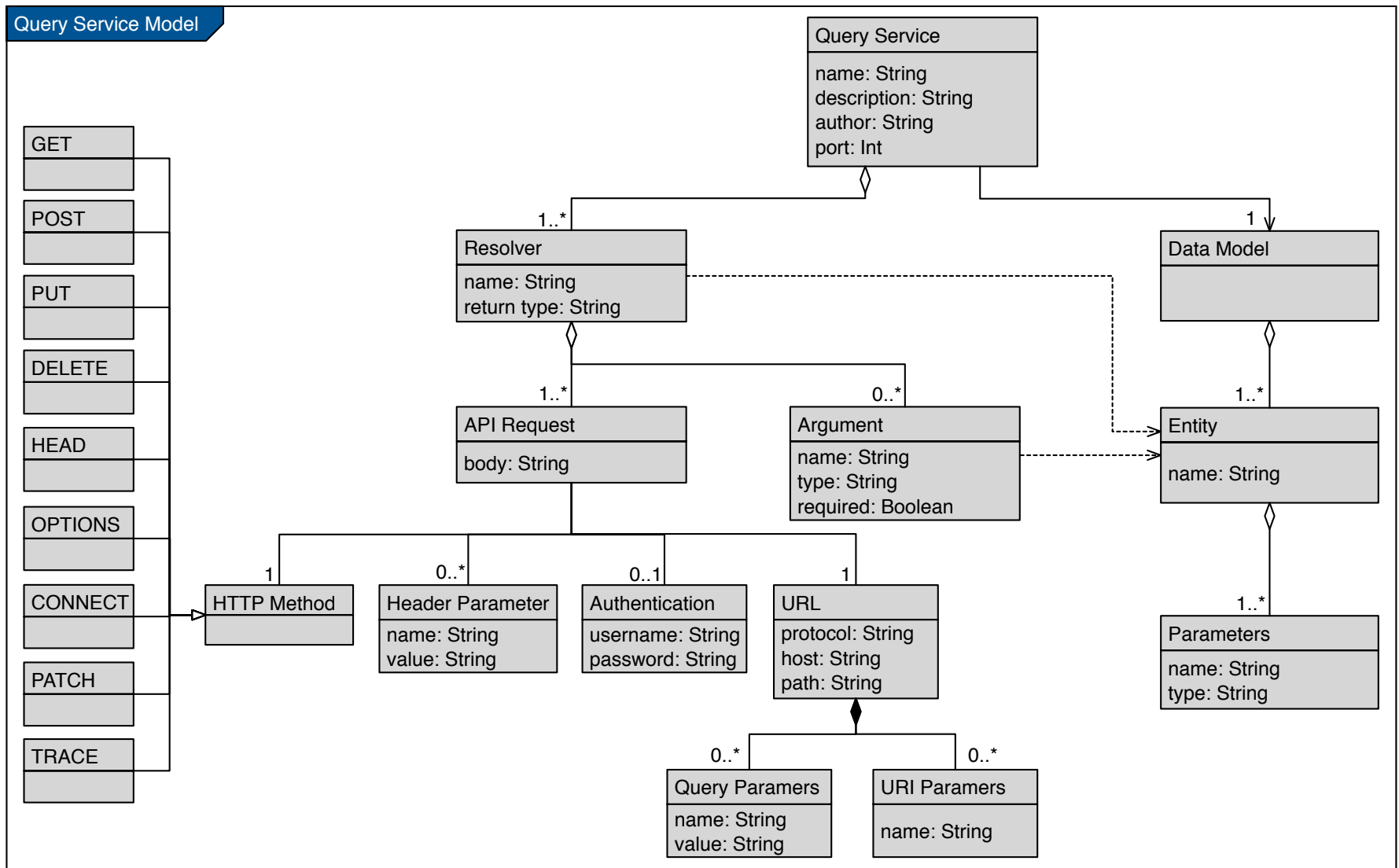
## Reference Architecture

- Architecture to reduce complexity in client
- shift responsibility of RESTful API consumption outside of client



## Meta-Model

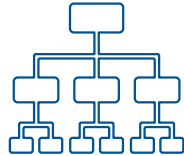
- Establishing meta-model describing consumption of RESTful services





## Reference Architecture

- Architecture to reduce complexity in client
- shift responsibility of RESTful API consumption outside of client



## Meta-Model

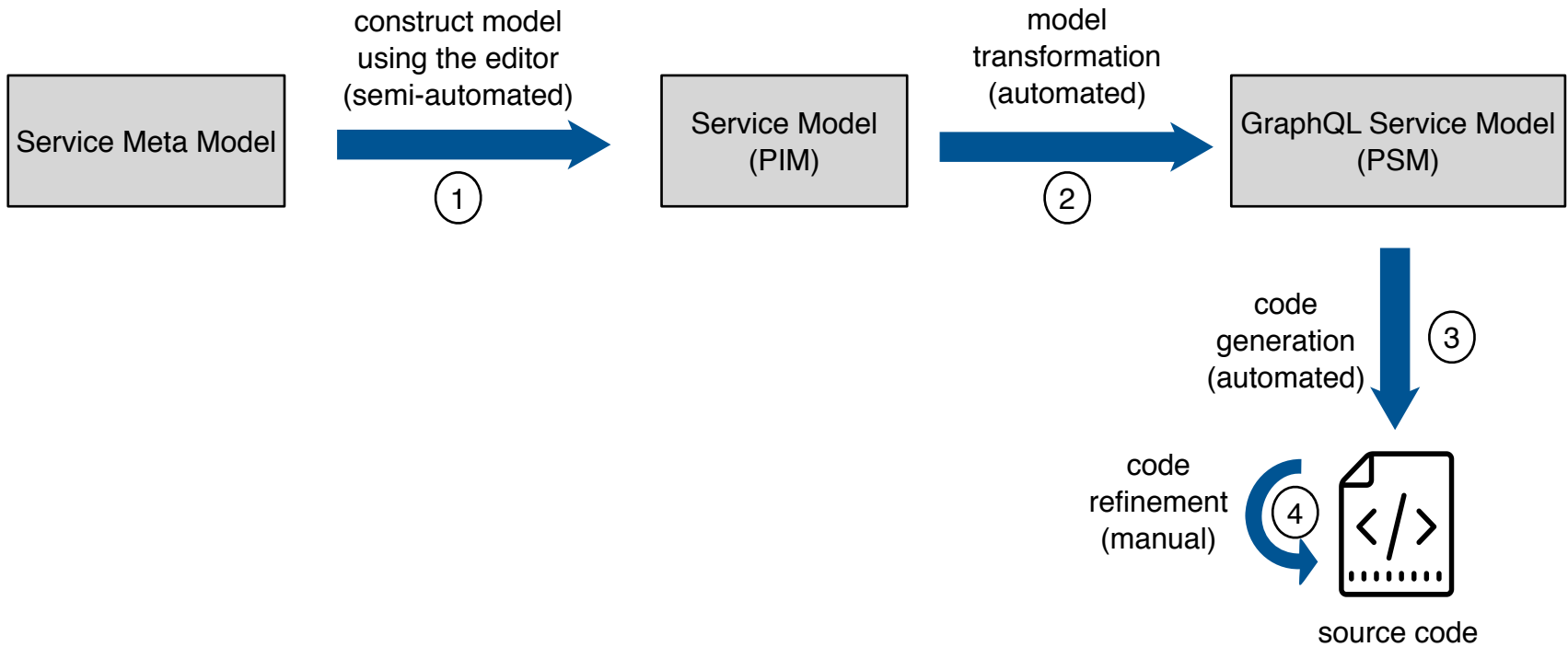
- Establishing meta-model describing consumption of RESTful services

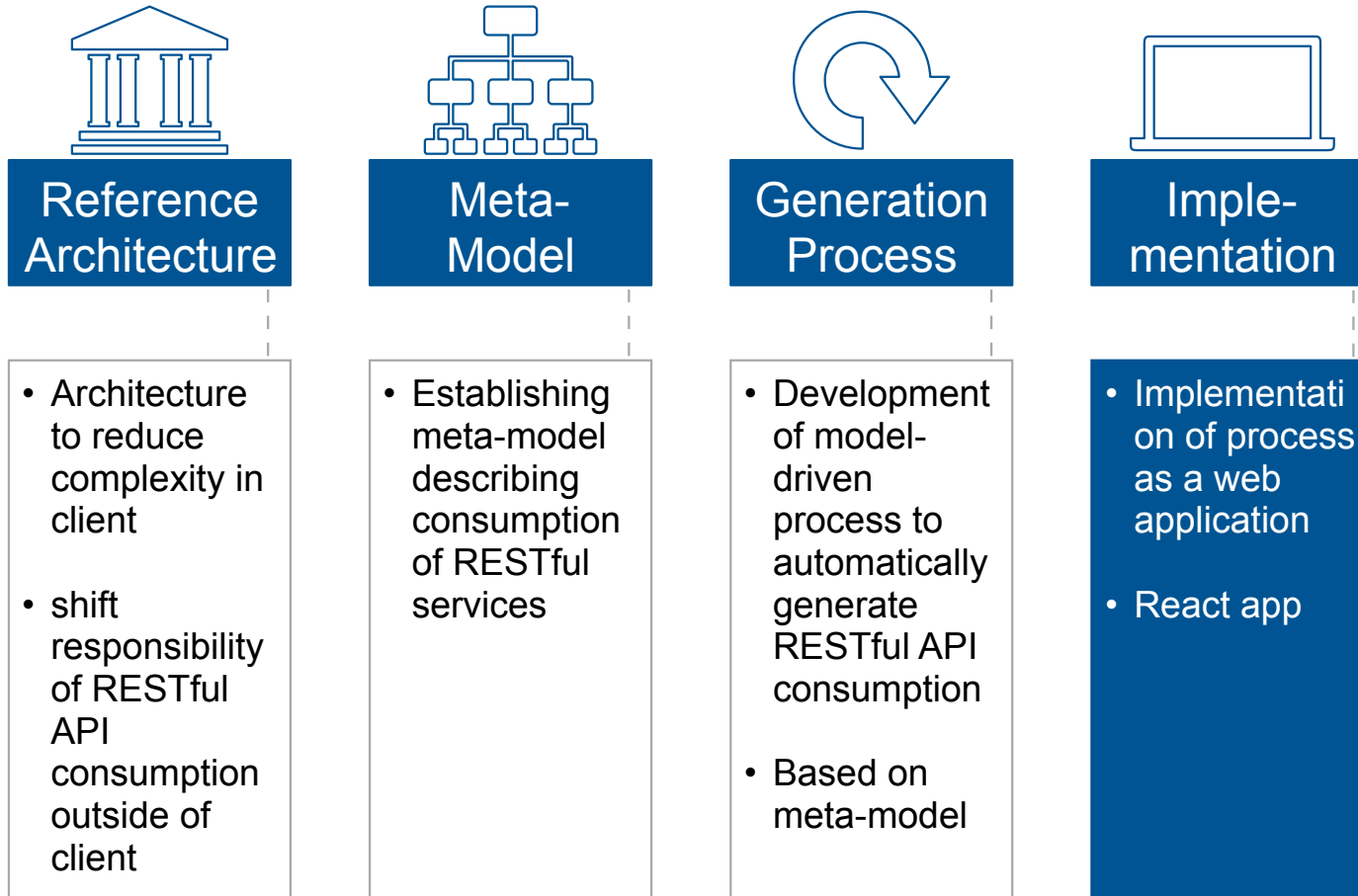


## Generation Process

- Development of model-driven process to automatically generate RESTful API consumption
- Based on meta-model

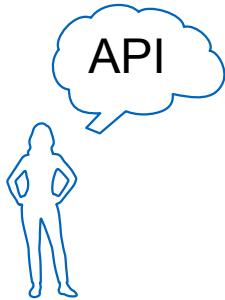
# Code Generation Process





# Code Generation Tool - Workflow

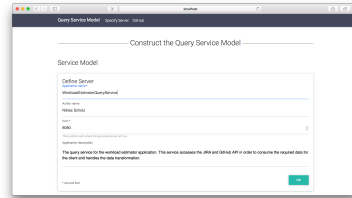
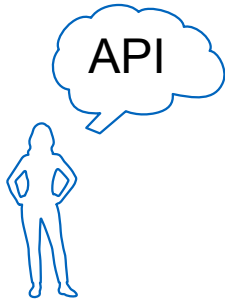
## Developer



1. What data and where to get it from?

# Code Generation Tool - Workflow

## Developer

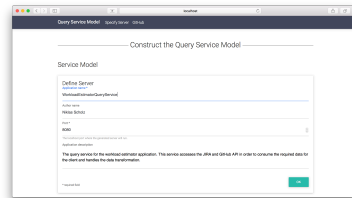
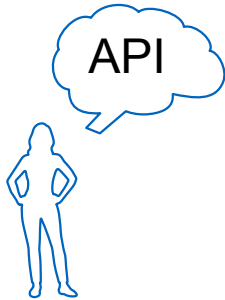


1. What data and where to get it from?

2. Construction of model

# Code Generation Tool - Workflow

## Developer



generation of  
query service



1. What data and  
where to get it from?

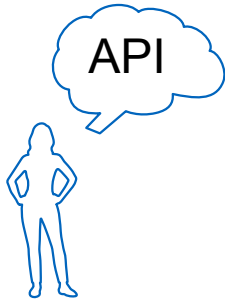
2. Construction  
of model

3. Manual code  
refinement

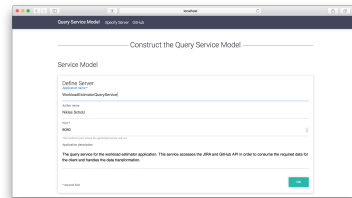


# Code Generation Tool - Workflow

## Developer



1. What data and where to get it from?



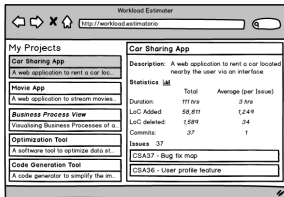
2. Construction of model

generation of query service



3. Manual code refinement

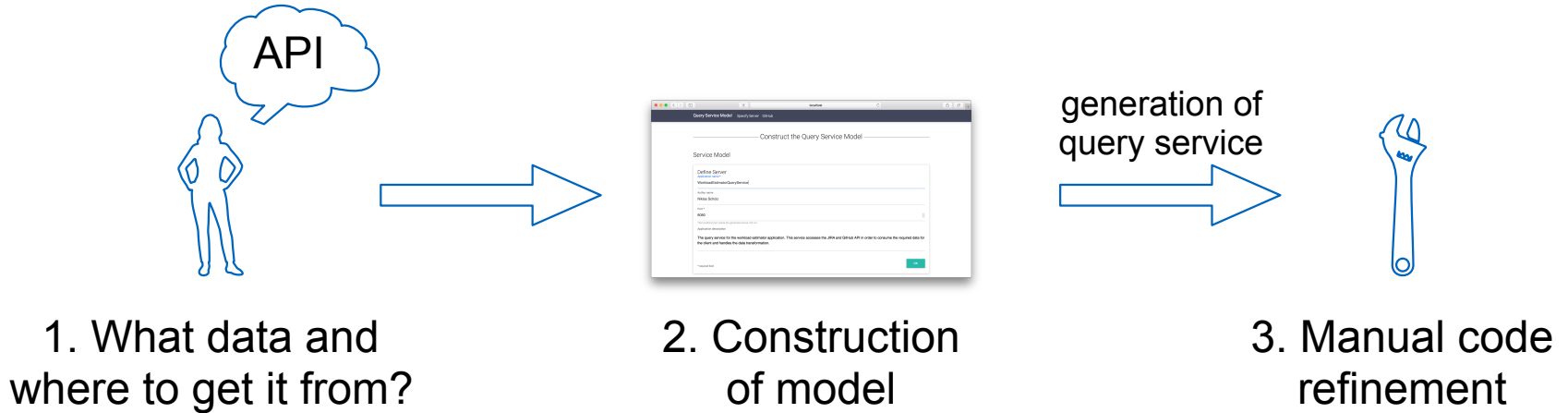
## UI Designer



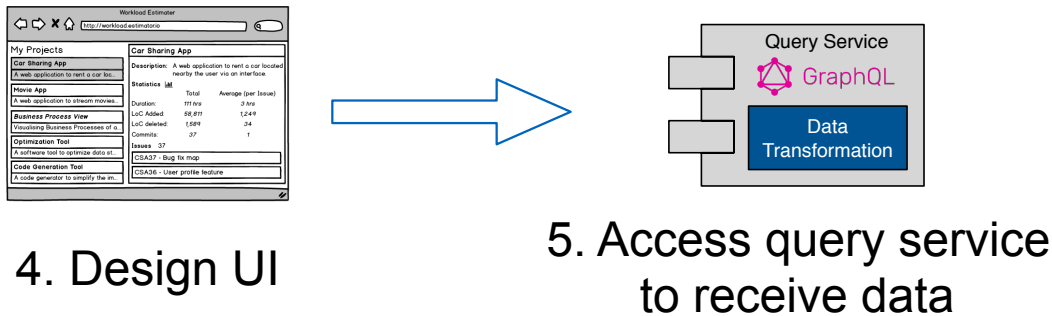
4. Design UI

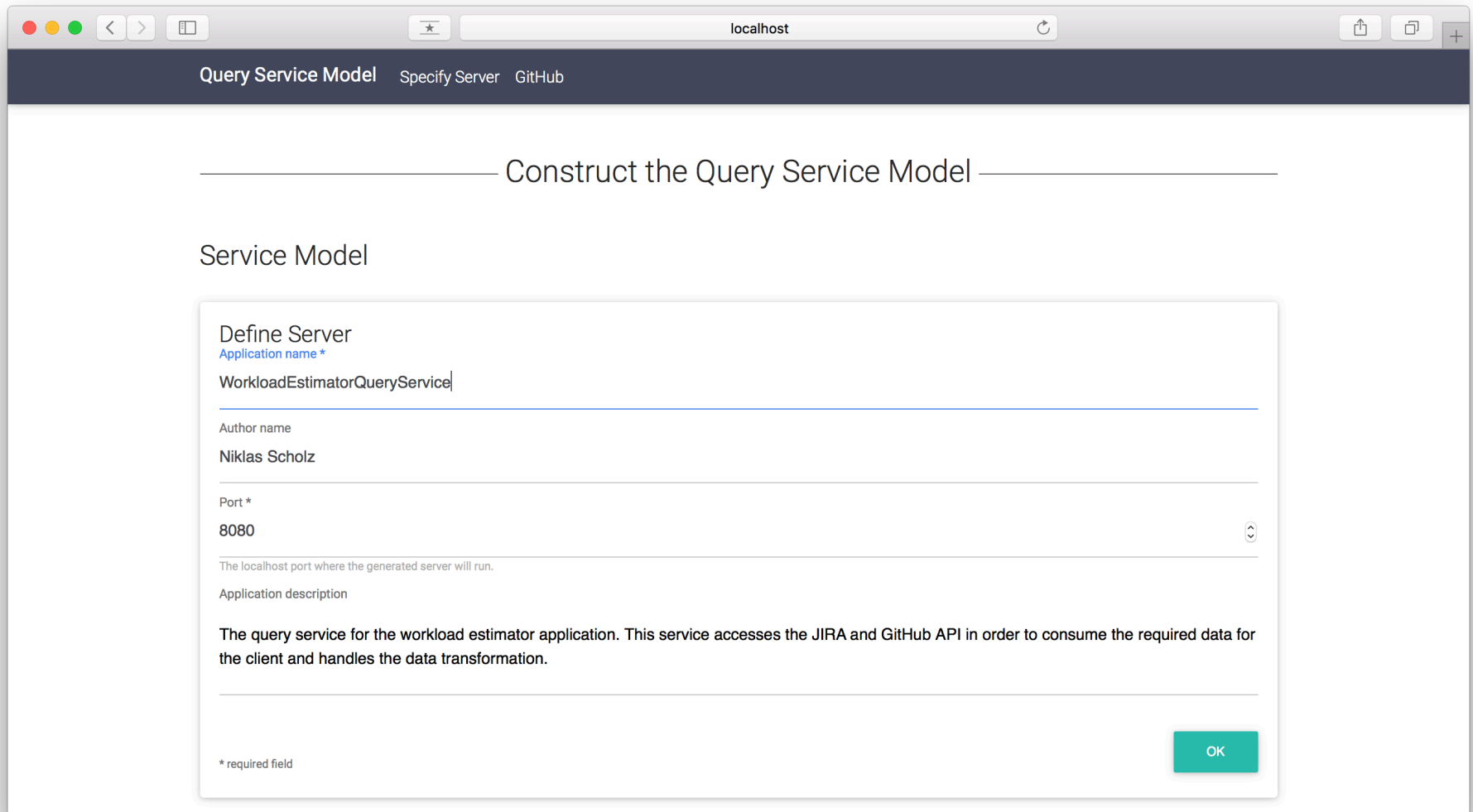
# Code Generation Tool - Workflow

## Developer



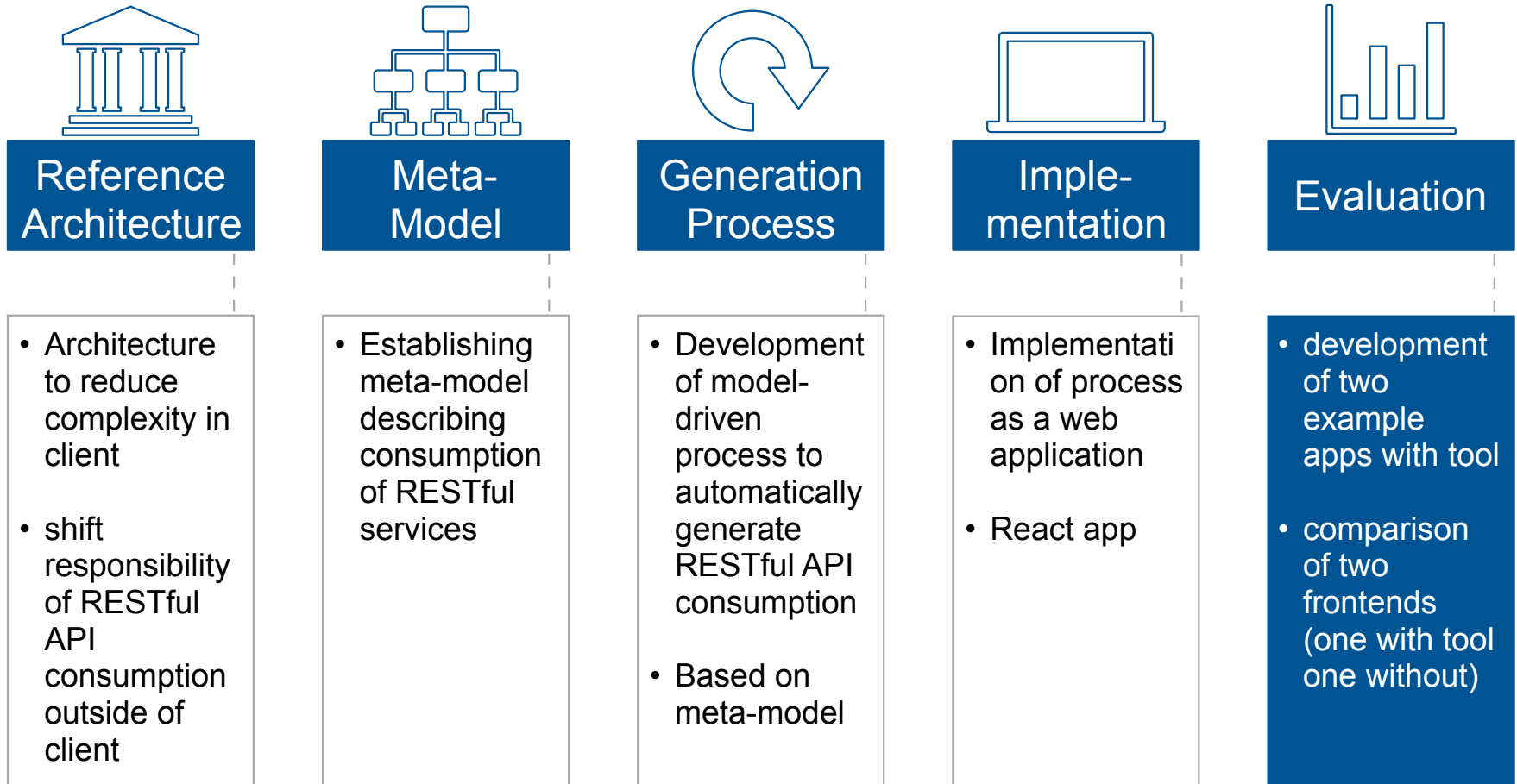
## UI Designer



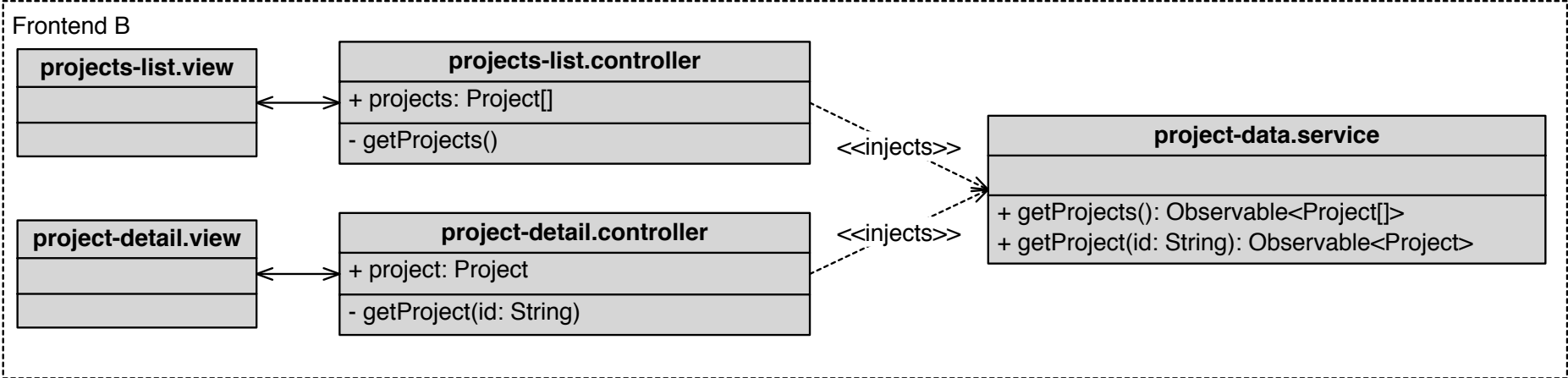
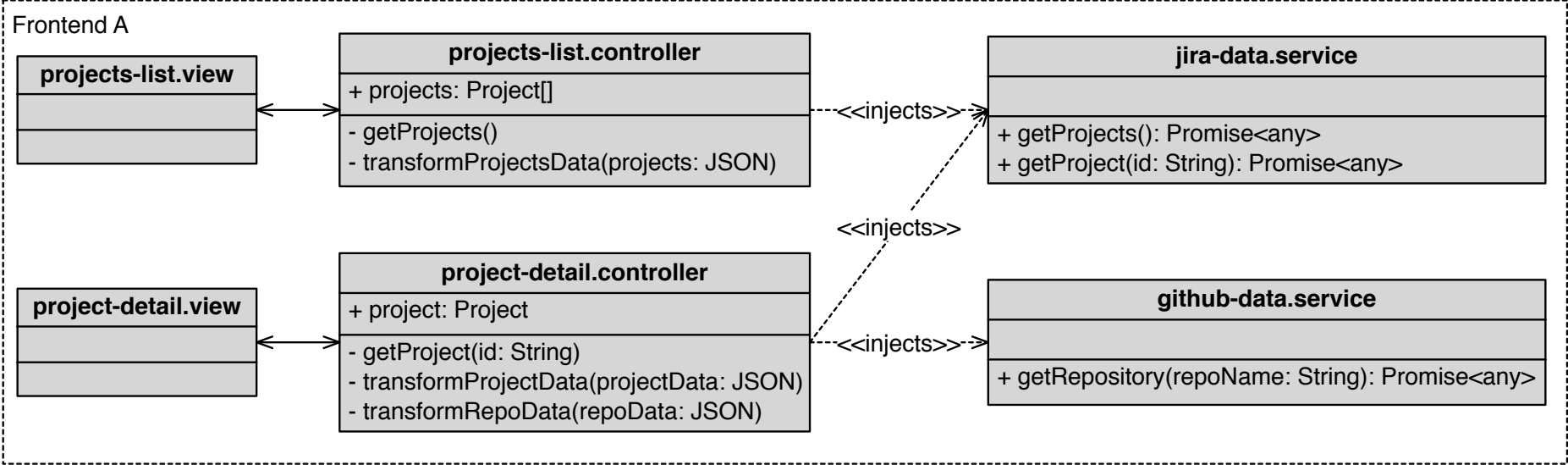


The screenshot shows a web browser window with the address bar set to 'localhost'. The browser's navigation bar includes 'Query Service Model', 'Specify Server', and 'GitHub'. The main content area features a heading 'Construct the Query Service Model' and a sub-section 'Service Model'. A modal dialog titled 'Define Server' is open, containing the following fields and text:

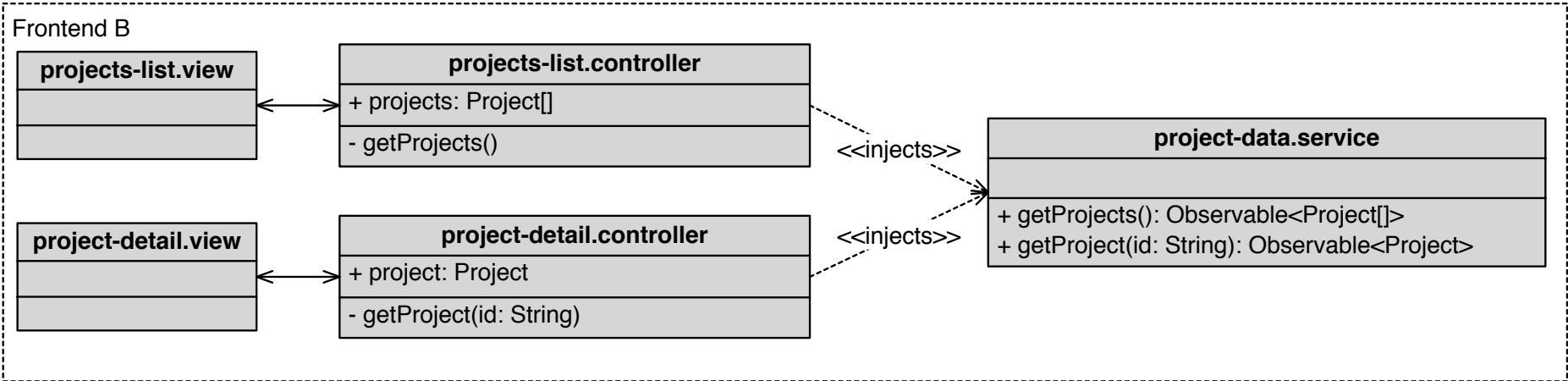
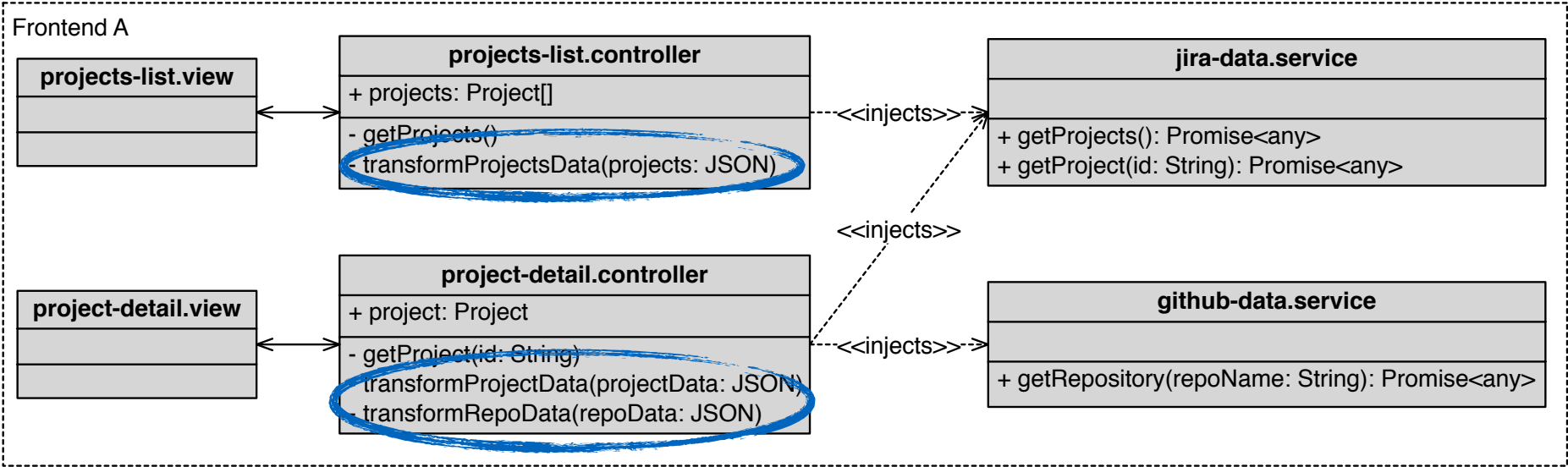
- Define Server**
- Application name \***: WorkloadEstimatorQueryService
- Author name**: Niklas Scholz
- Port \***: 8080
- The localhost port where the generated server will run.
- Application description**: The query service for the workload estimator application. This service accesses the JIRA and GitHub API in order to consume the required data for the client and handles the data transformation.
- \* required field
- OK** button



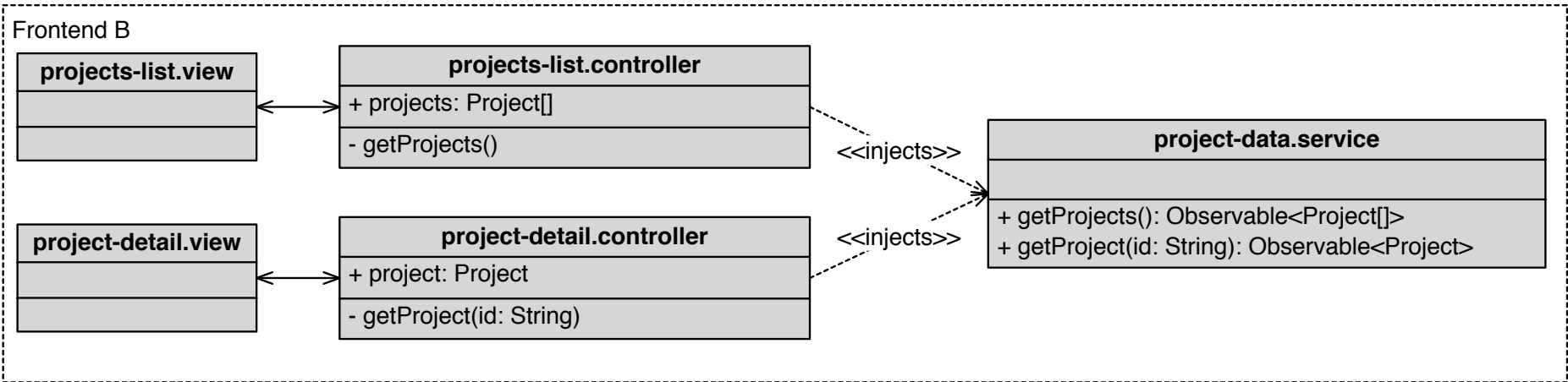
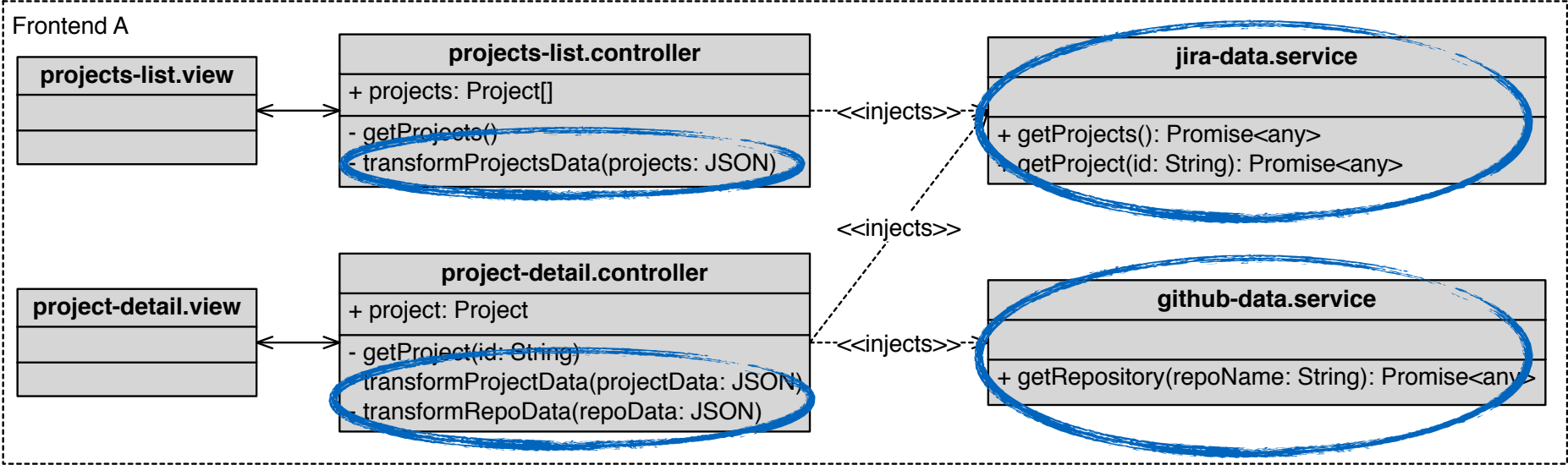
# Evaluation - Comparing Two Frontends



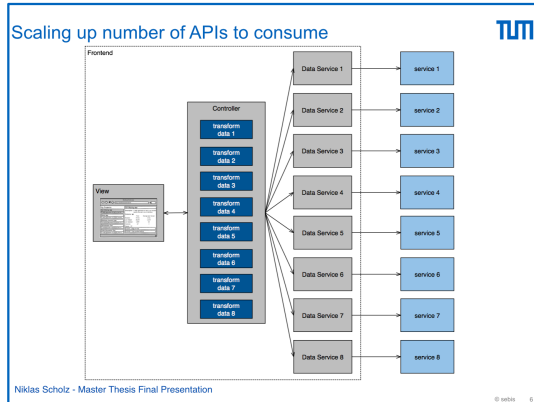
# Evaluation - Comparing Two Frontends



# Evaluation - Comparing Two Frontends



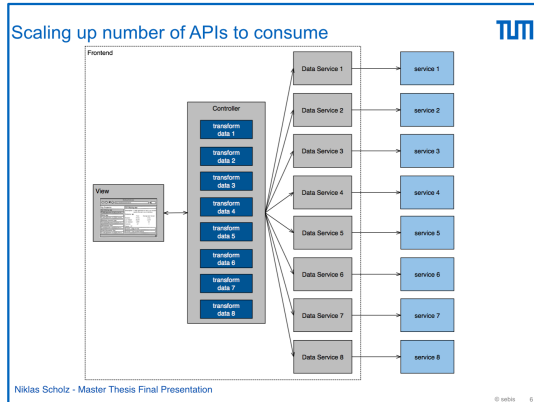
# Conclusions



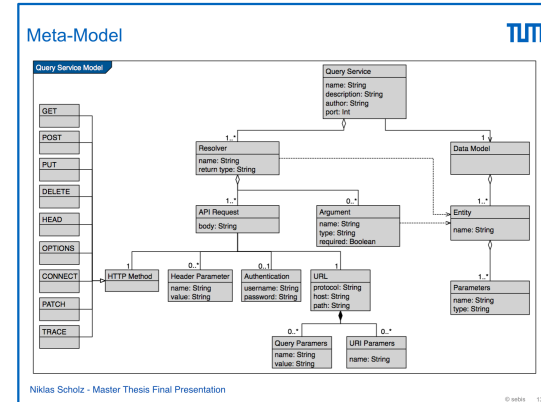
Consuming multiple APIs leads to complexity in the client



# Conclusions

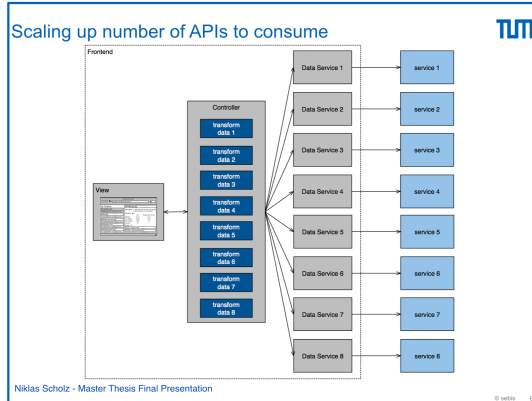


Consuming multiple APIs leads to complexity in the client

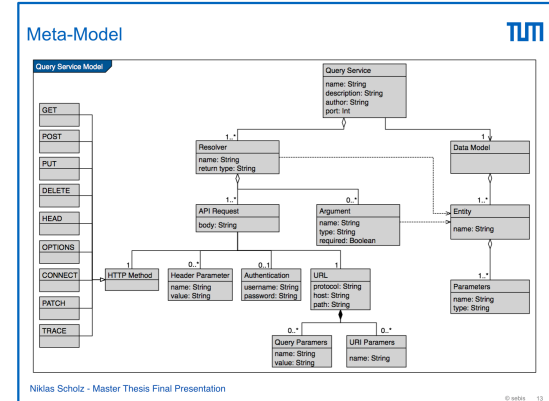


The consumption of RESTful services can be modelled

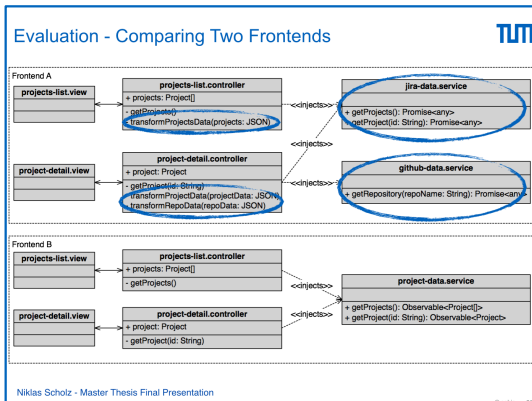
# Conclusions



Consuming multiple APIs leads to complexity in the client

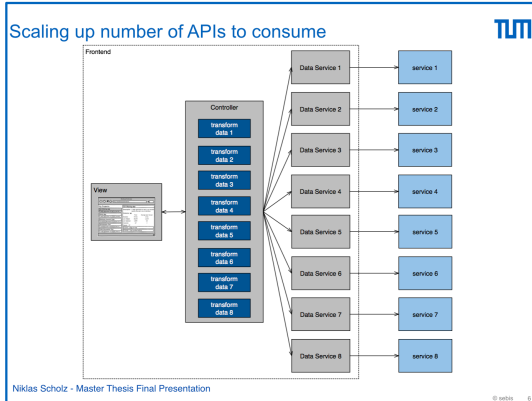


The consumption of RESTful services can be modelled

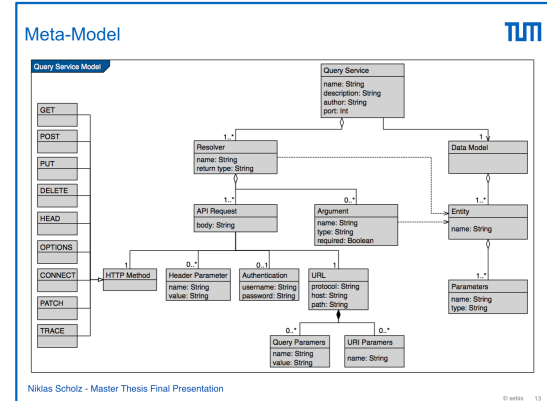


Approach reduces complexity in client

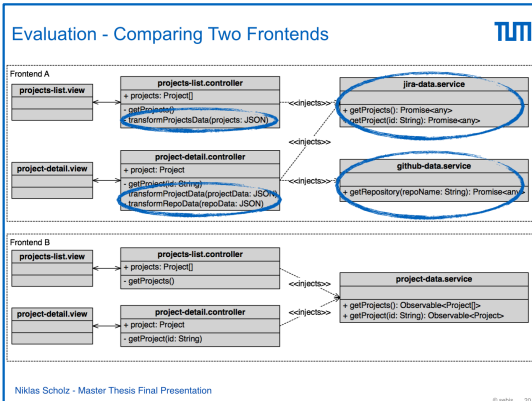
# Conclusions



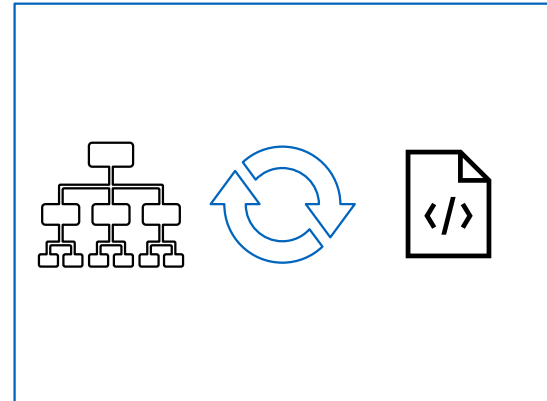
Consuming multiple APIs leads to complexity in the client



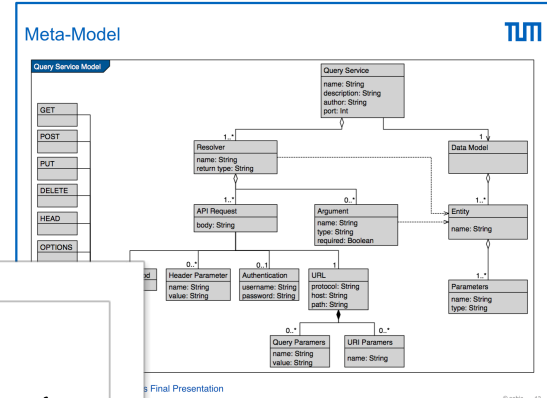
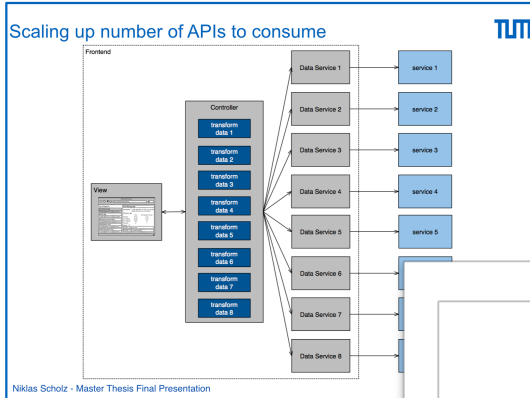
The consumption of RESTful services can be modelled



Approach reduces complexity in client



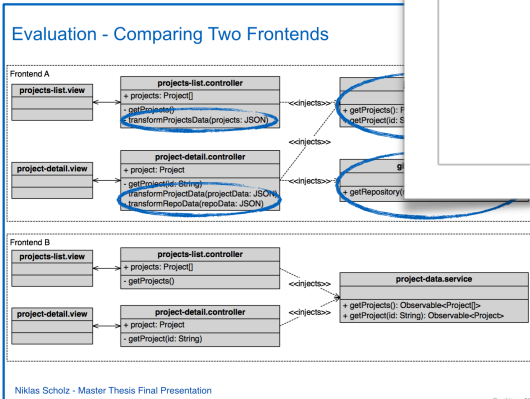
Limitation: Roundtrip Engineering



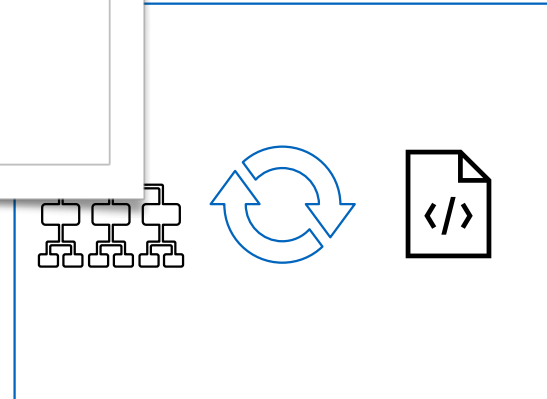
Consuming multiple API complexity in the

**THANKS!**

consumption of RESTful services can be modelled



Approach reduces complexity in client



- [1] Markus Lanthaler and Christian Gütl. ‘Hydra: A vocabulary for hypermedia-driven web APIs’. In: *CEUR Workshop Proceedings 996* (2013). issn: 16130073.
- [2] Davide Rossi. ‘UML-based Model-Driven REST API Development’. In: *Proceedings of the 12th International Conference on Web Information Systems and Technologies, Vol 1 (WEBIST)* (2016), pp. 194–201. doi: 10.5220/0005906001940201.
- [3] Hamza Ed-Douibi et al. ‘EMF-REST: Generation of RESTful APIs from Models’. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing* (2015), pp. 1446–1453. doi: 10.1145/2851613.2851782.
- [4] Florian Haupt et al. ‘A model-driven approach for REST compliant services’. In: *Proceedings - 2014 IEEE International Conference on Web Services, ICWS 2014* (2014), pp. 129–136. doi: 10.1109/ICWS.2014.30.
- [5] Rodrigo Bonifacio et al. ‘NeoIDL: A Domain-Specific Language for Specifying REST Services’. In: *International Conference on Software Engineering and Knowledge Engineering* (2015), pp. 613–618. doi: 10.18293/SEKE2015-218.

## Lightweight Client

- Reducing complexity on client-side
- GraphQL queries

## Focus on important steps

- not carried away with implementation details
- Focus on UI

## Advantages of MDS

- consistency
- reusability
- development speed
- manageability of complexity

## Roundtrip Engineering

- Model not being stored
- Apply API changes manually

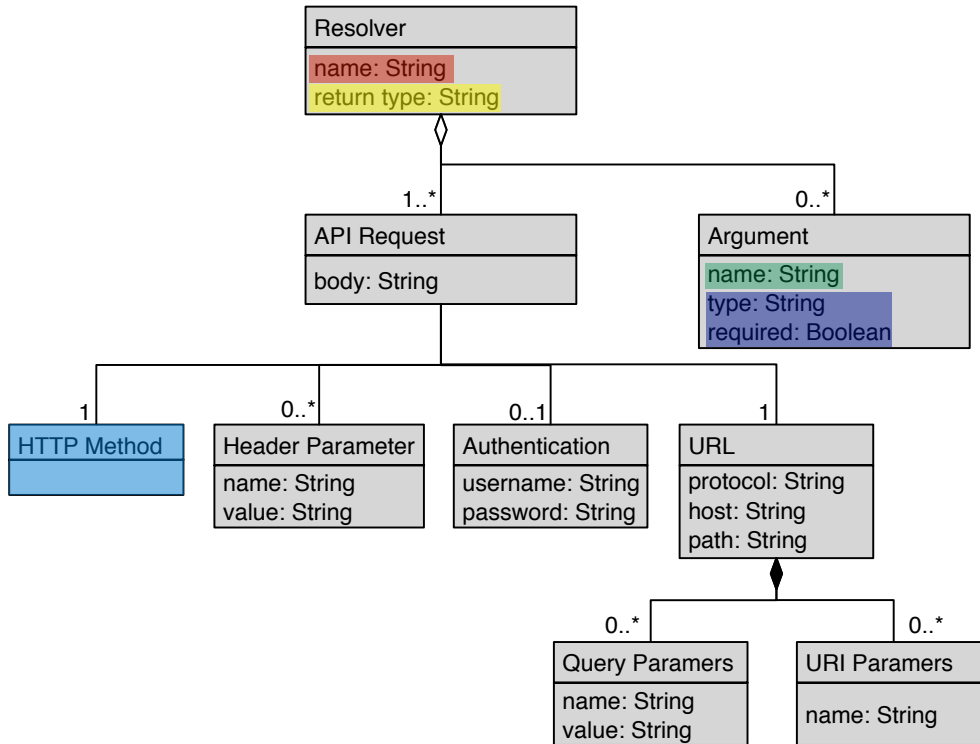
## Manual Code Refinement

- Data transformation not modelled
- Developer has to add code manually

## Level of Abstraction

- only RESTful APIs
- model not entirely generic

Resolver Model



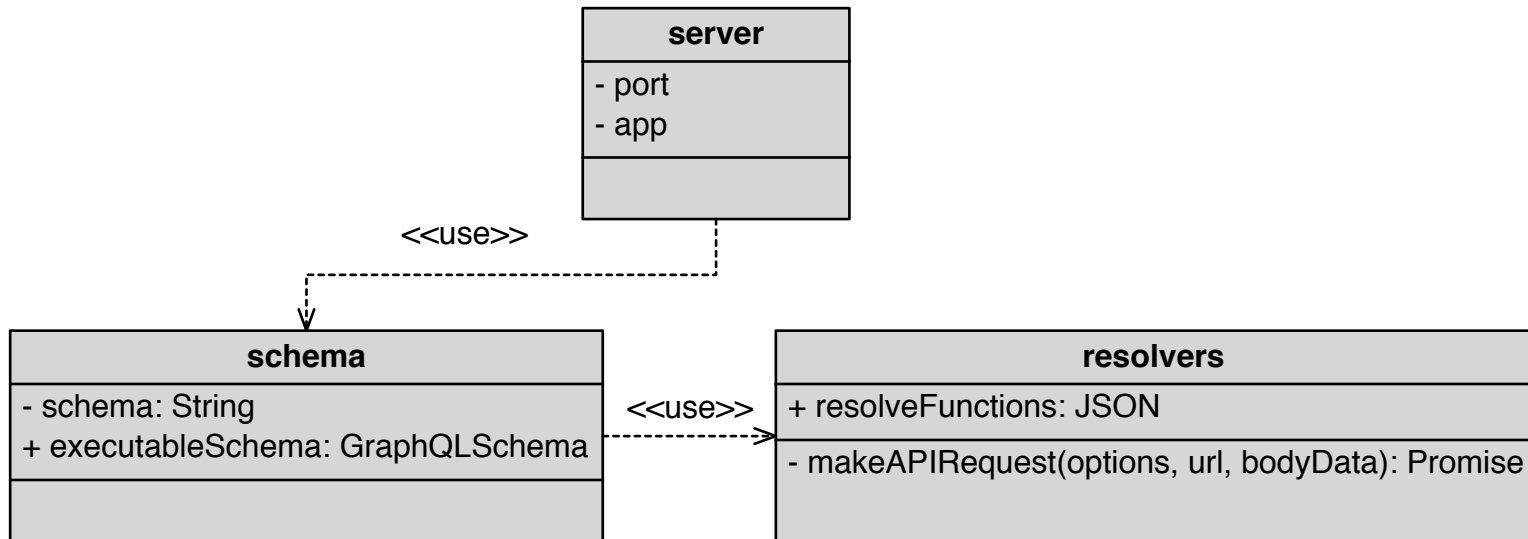
GraphQL Schema

```
#the schema allows the following query:
type Query {
  projects: [Project]
  project(projectId: String!): Project
}
```

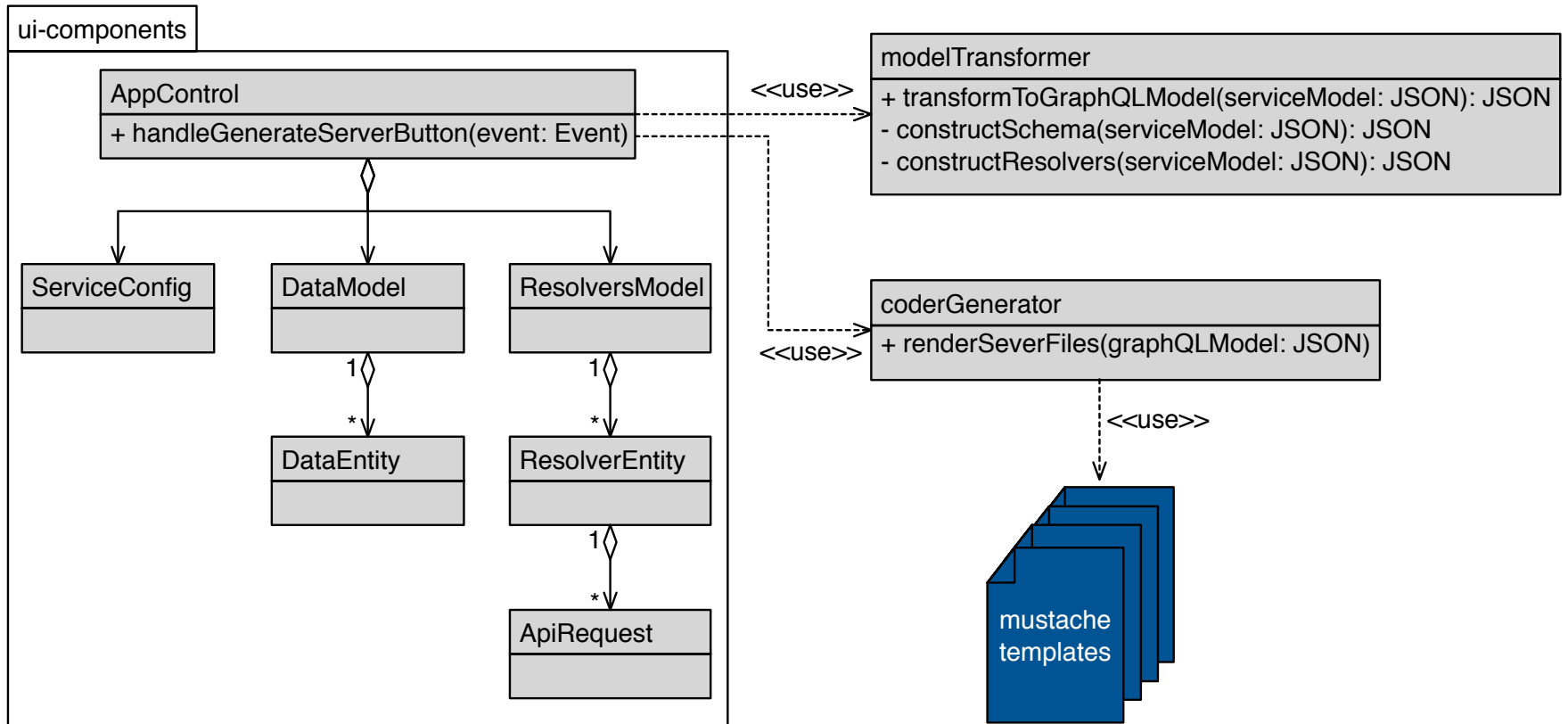
```
#the schema allows the following mutations:
type Mutation {
  deleteProject(projectId: String!): Project
}
```



# Backup - Query Service Architecture



# Backup - Code Generation Tool Architecture



# Backup - GraphQL Schema Example



```
type Project {
  projectId: String
  name: String
  description: String
  duration: Float
  nrCommits: Int
}

#the schema allows the following query:
type Query {
  projects: [Project]
  project(projectId: String!): Project
}

#the schema allows the following mutations:
type Mutation {
  deleteProject(projectId: String!): Project
}
```