How and why?

redpencil.io

Agenda

What is semantic.works?

What makes it tick?

Extensions and future work

In 60 seconds

State-of-the-art web applications fueled by Linked Data aware microservices

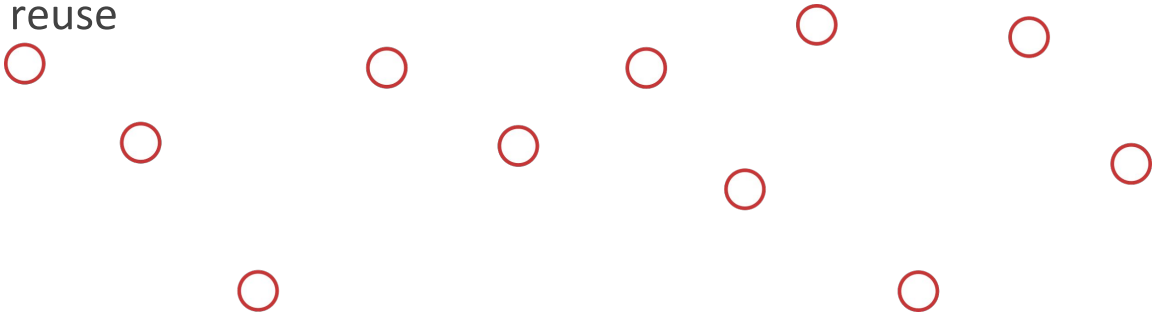
- User-facing microservices
- Easy deployment using Docker
- Single Page Apps using Ember.js
- Well known requirements

=> [HTTP+JSON+SPARQL]

microservices birds-eye

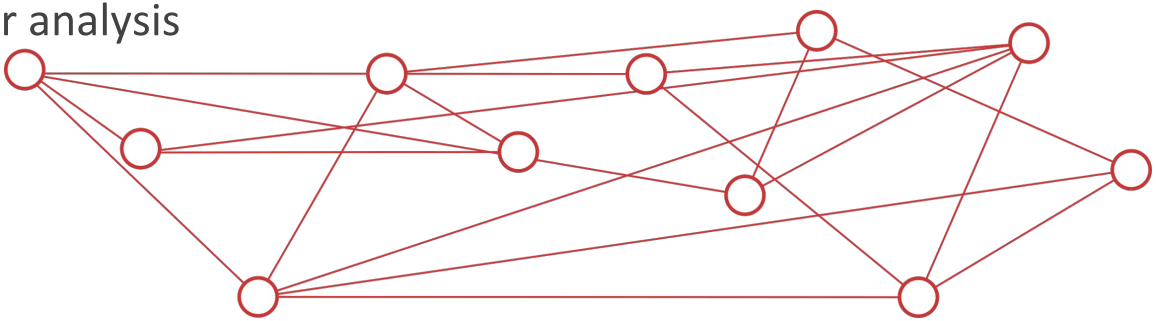
Microservices are awesome

- Easy to understand
- Easy to debug
- Easy to reuse



Microservices are complex

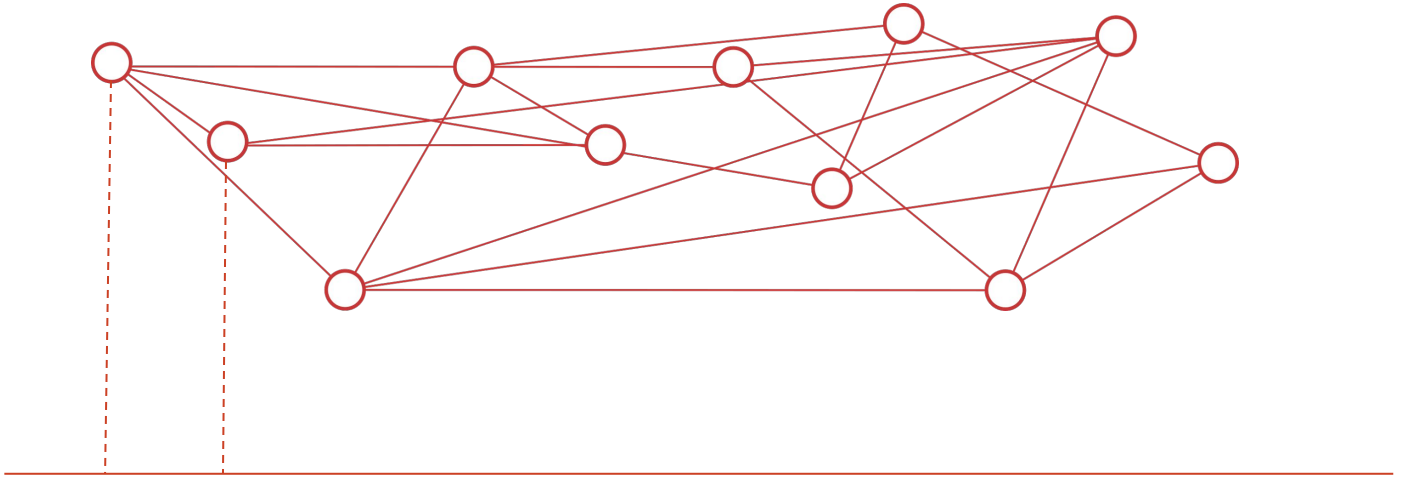
- Data model dependencies
- API dependencies
- Disaster analysis



Microservices are to be reengineered

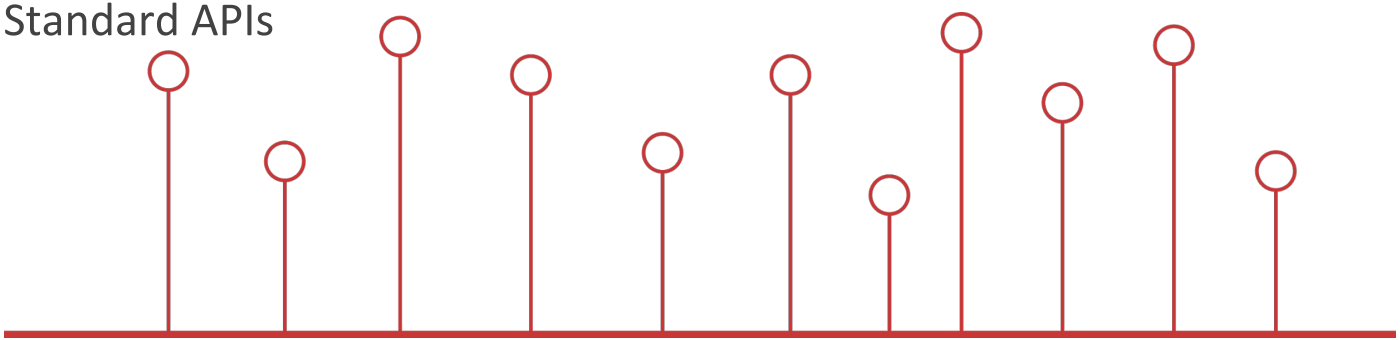
- Direct connection to the database

- Using semantic modelling

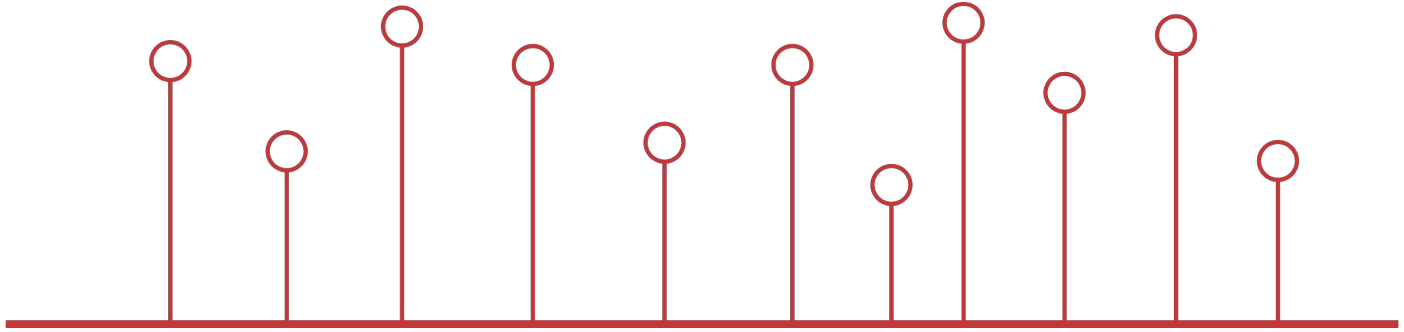


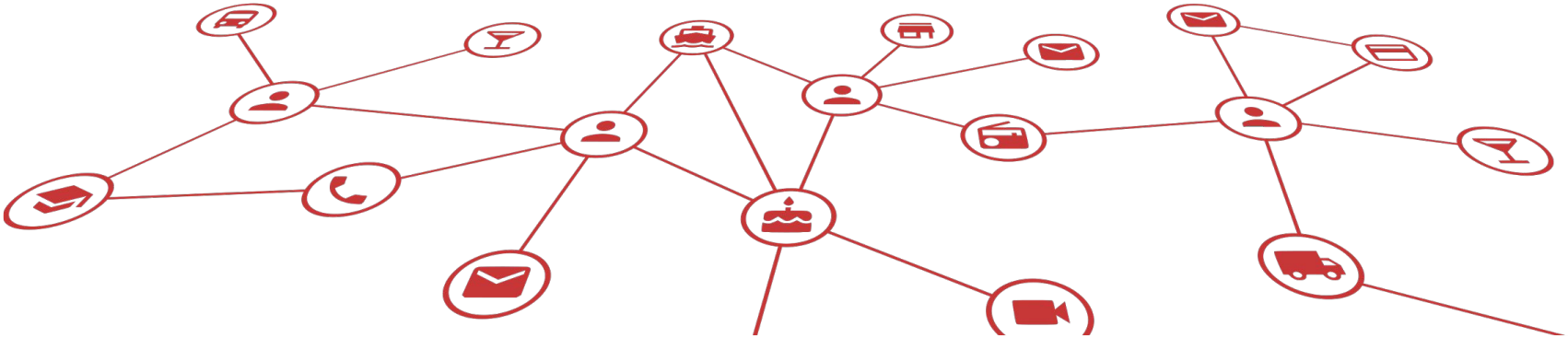
Microservices are micro standalone services

- Embrace Semantic Model
- Functional microservices
- Standard APIs



... taking advantage of the semantic domain model



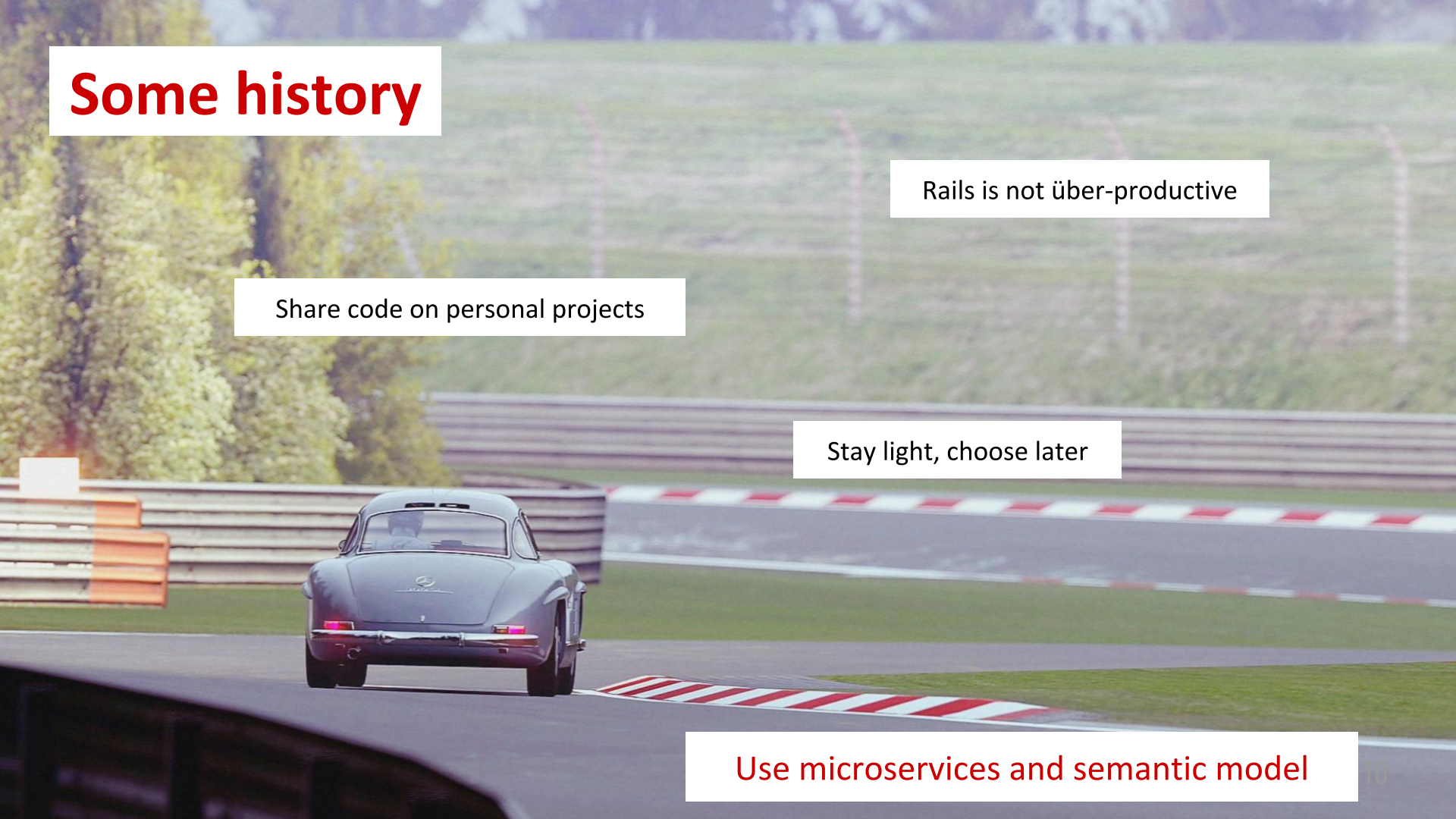


Some history

Rails is not über-productive

Share code on personal projects

Stay light, choose later



Use microservices and semantic model

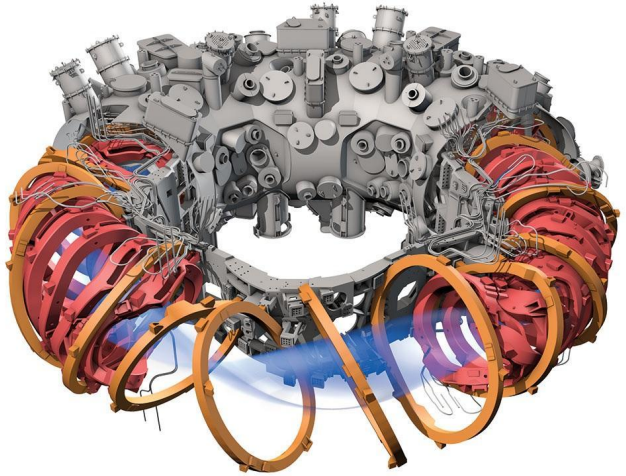
semantic.works values

KISS

Keep It Super Simple

KISS

- Most of us aren’t microservice experts
- Most of us aren’t UI experts
- We need to get stuff done

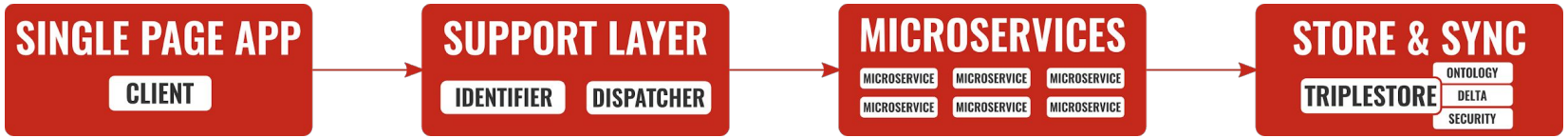


- Maximize freedom
- Orthogonal features
- Minimize requirements
- Enforce simple mental model

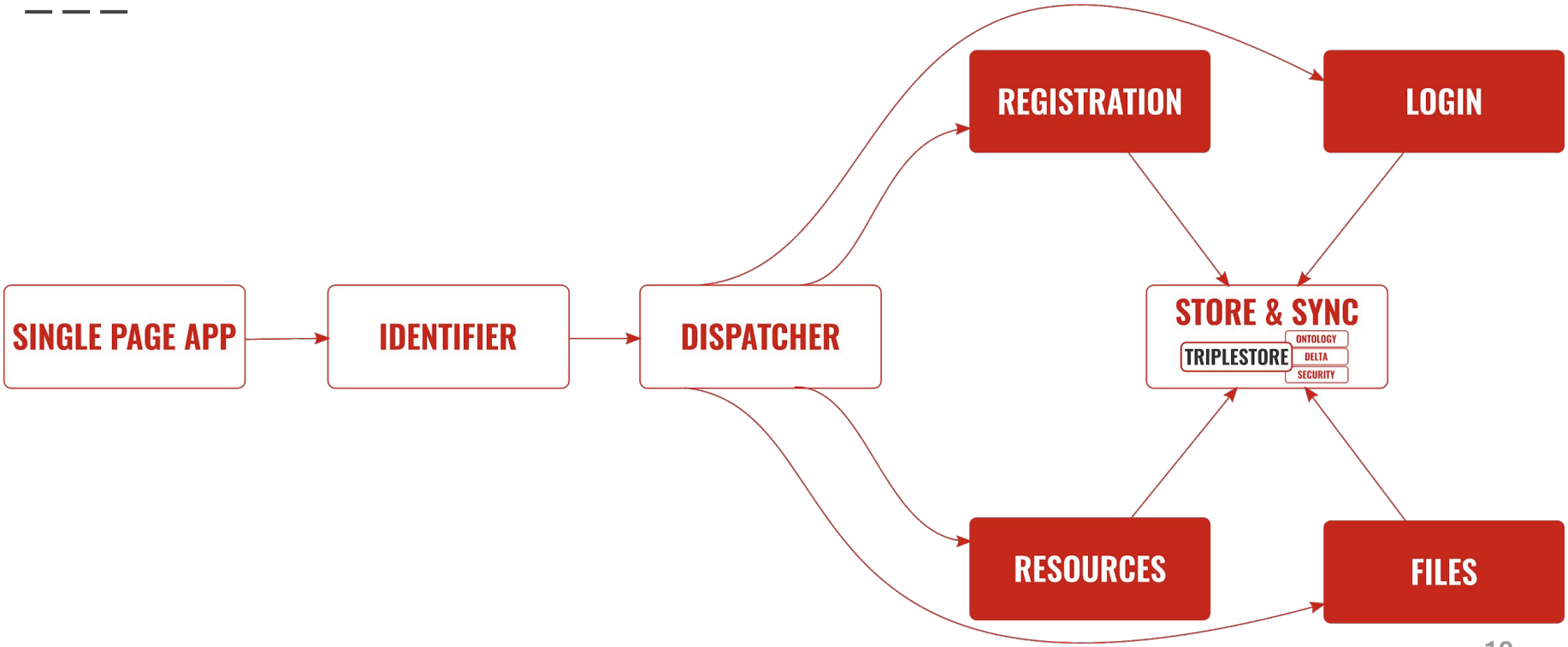
Simple mental model

user-facing http-services

Simple mental model

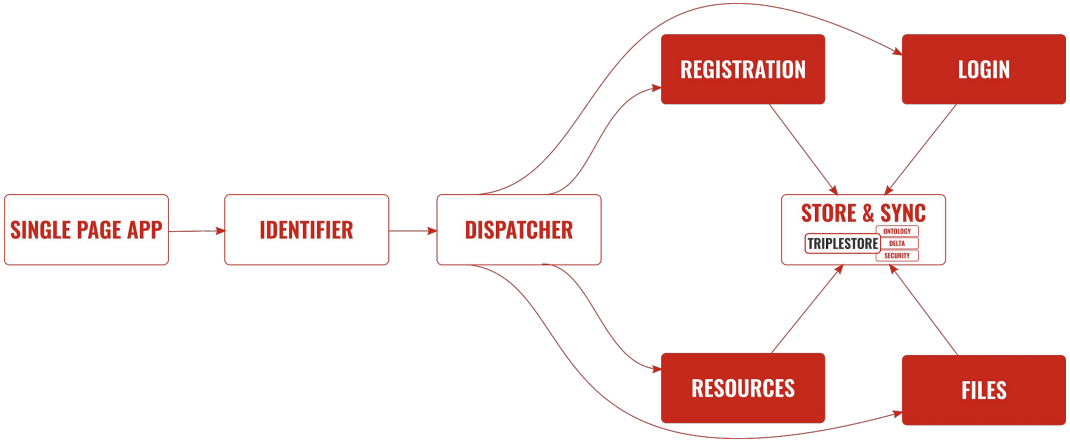


Simple mental model



Simple mental model

User-facing microservices



Limit base technologies:

- HTTP
- JSON(API)
- SPARQL (one graph)

Semantic models

Many actors, telling parts of the same story

Semantic models

Services read/write the part of the world they understand.

User Registration:

- There’s a new user => add it to the triplestore.

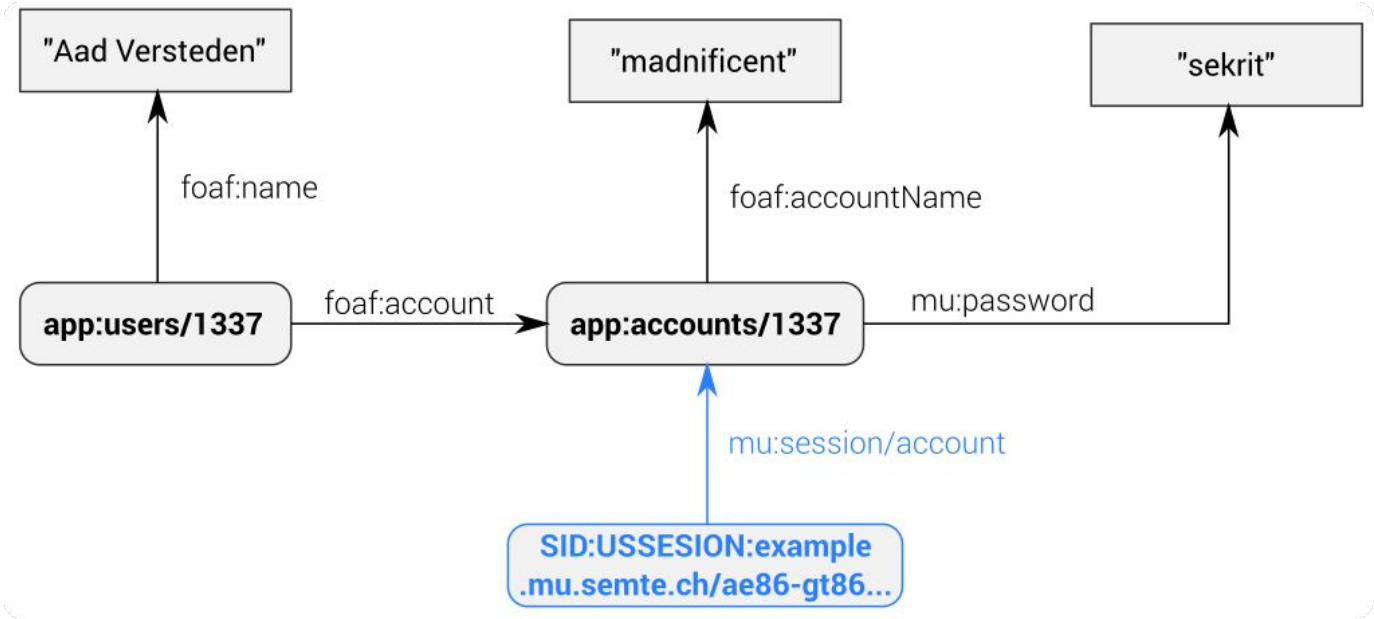
User Login:

- Check username/password => connect user to current session.

Semantic models

Registration service

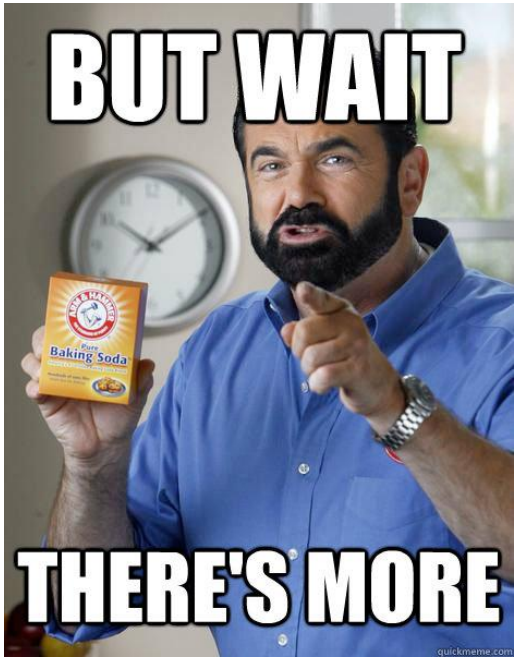
Login service



Semantic models

Same model for:

- Username/Password login
- OAuth login
- ACM/IDM login



Many implementations, one model

Docker

Deployment made easy

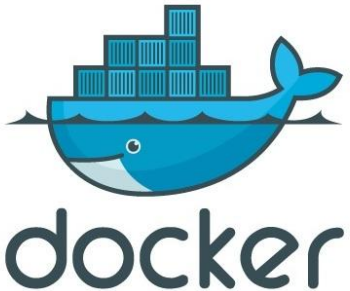
Share using Docker

Docker Container =~ Lightweight Linux Virtual Machine
Docker Compose =~ Topology of multi-container project

Each service runs in its own Docker Container

In short:

- Simple hosting on hub.docker.com
- Clean project description
- Always works



Reuse everything

Reuse everything

- Templates: basics for a service

- Configurable services
  - mu-cl-resources

- Ember add-ons
  - Data table addon
  - Login add-on

Base templates

[mu-ruby-template]

```
get '/hello/' do
  counter = query( "SELECT COUNT (*) as ?counter" +
                   "WHERE {" +
                   "  ?s ?p ?o." +
                   "}" ).first[:counter].to_i
  status 200
  { value: counter }.to_json
end
```

```
FROM semtech/mu-ruby-template:2.0.0-ruby2.3
MAINTAINER Your Name <you@provider.com>
```

```
CatalogsIndexRoute = Ember.Route.extend
 ajax: Ember.inject.service()
 model: () ->
   @get('ajax').request('/hello')
```

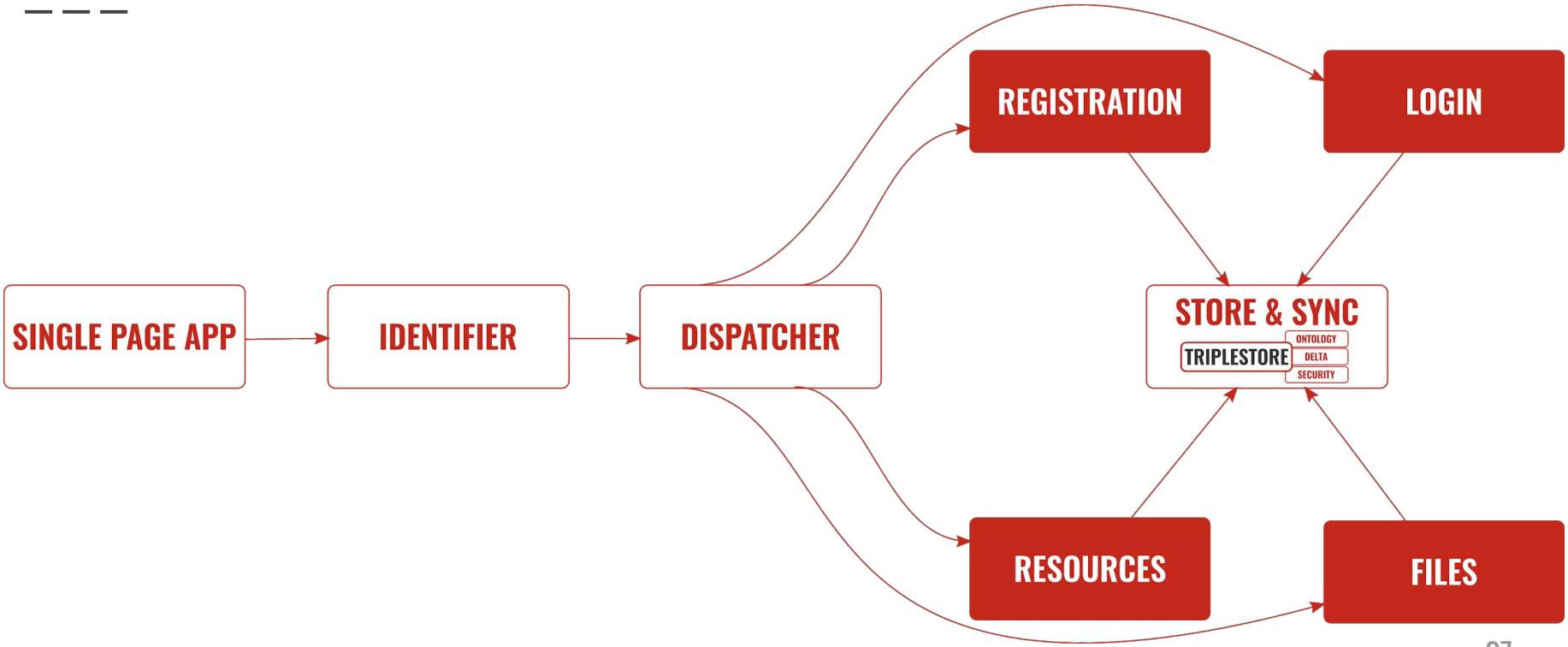
```
Hello result: {{model.value}}
```

Hello result: 5841

```
demo:
  image: you/demo-service
  links:
    - db:database
dispatcher:
…
links:
  - demo:demo
```

```
match "/hello/*path" do
  Proxy.forward conn, path, "http://demo/hello/"
end
```

Base templates



Configurable services

[mu-cl-resources]

Full JSONAPI from abstract description

```
{ "prefixes": [
    { "dcat": "http://www.w3.org/ns/dcat#" }
  ],
  "resources": {
    "datasets": {
      "class": "dcat:Dataset",
      "new-resource-base": "https://example.com/datasets/",
      "attributes": {
        "title": {
          "type": "string",
          "predicate": "dct:title"
        }
      },
      "relationships": {
        "catalog": {
          "target": "catalogs",
          "predicate": "dcat:dataset",
          "inverse": true,
          "cardinality": "one"
        },
        "themes": {
          "target": "theme",
          "predicate": "dcat:theme",
          "cardinality": "many"
        }      }    }  }}
```

Ember add-ons

```
 > ember install ember-paper-data-table

 // template
{{data-table
     content=model
     fields="title isbn genre publicationDate language numberOfPages"
     sort=sort page=page size=size}}

 // route
import Ember from 'ember';
import DataTableRouteMixin from 'ember-data-table/mixins/route';

export default Ember.Route.extend(DataTableRouteMixin, {
  modelName: 'book'
});
```

Semantic Works -- extra

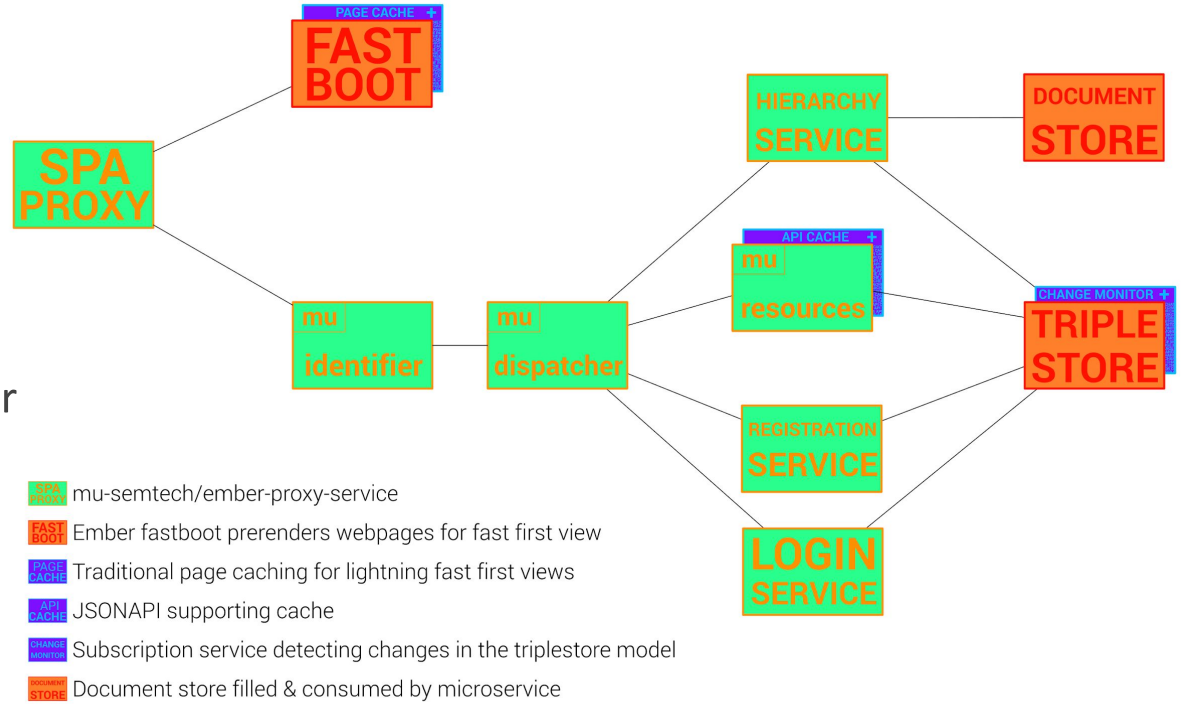
Extensions made to the base concept

More performance

- ember-fastboot:
  Faster first page render
  Future: cache it!

- mu-cache:
  Smart caching strategies for
  core microservices

- mu-cl-resources:
  Partial resource caching



More authority

Describe authorization outside the microservices:

- Simplify mental model
- Help in sharing content
- Opens gate to advanced applications

Reactive programming

Trigger microservices by changes in semantic model.

Examples:

- Send email/tweet by writing it to the triplestore
- Compute KPIs when a new dataset is added

Shell command

Automate routine tasks

Generate new service

Wire up new microservices

Extensible with custom commands per microservice

Semantic Works -- future

Future and ongoing efforts to improve dev time

Future: documentation flow

Generate browsable documentation for your semantic.works application.

Documentation of semantic model on which services acts.

Future: interactivity

Push cache updates to all visiting clients.

Almost no development time to create basic interactive applications

(eg: updating KPIs, chat applications, …)

Future: live replica

Full live replication of database on remote location.

Partial replication of data between applications.

Let's throw some love at it.