22.03.2018

A microservice approach in the boundaries of a traditional enterprise environment

Name: Sebastian Eggers

Conference: microXchg 2018

Location: Berlin (Germany)

Who am I ?

Working at E.ON since 2010

Based in Hanover

Passionate Developer

Focus areas: APIs, Microservices, Architecture

Teamlead Software Development

Agenda

1. Constraints & Motivation for „Microservice Blueprint“
2. Details of „Microservice Blueprint“
3. Living Project Example
4. Summary

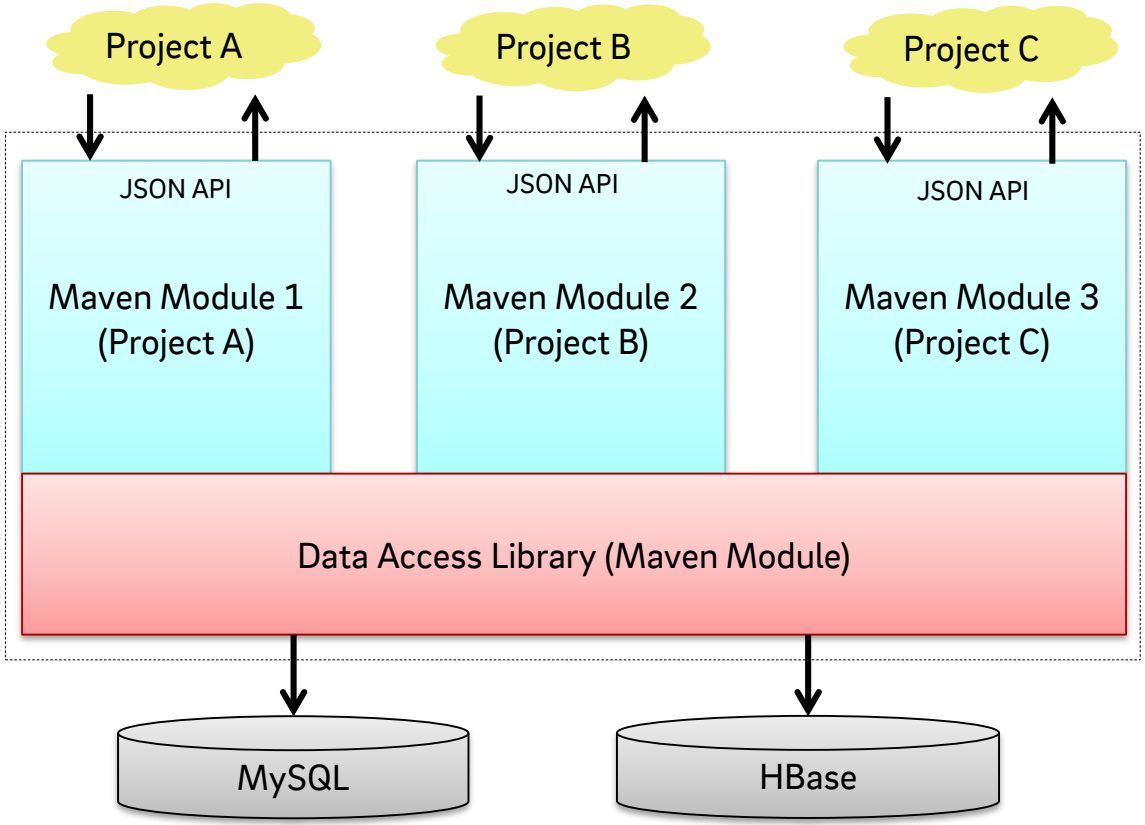
Constraints & Motivation for „Microservice Blueprint“

I was asked for a code review and found this ...

```
public interface ApplianceDAO {

    @SqlQuery("select * from appliances where owner_id = :owner_id")
    @Mapper(ApplianceMapper.class)
    List<Appliance> findAllAppsofOwner(@Bind("owner_id") int owner_id);

    //@SqlUpdate("CREATE TABLE IF NOT EXISTS appliances (id INT(11) NOT NULL AUTO_INCREMENT, name TINYTEXT NOT NULL, quantity TINYINT(3) UNSIGN
      @SqlUpdate("CREATE TABLE `appliances` ("+
      "`id` INT(11) NOT NULL AUTO_INCREMENT,"+
      "`quantity` TINYINT(3) UNSIGNED ZEROFILL NOT NULL DEFAULT '000',"+
      "`appliance_map_id` INT(11) NOT NULL,"+ "`name` TINYTEXT NULL,"+
      "`utc_ts` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,"
      + "`owner_id` INT(11) NOT NULL,"+ "PRIMARY KEY (`id`),"+
      "INDEX `FK1_owners` (`owner_id`),"+
      "INDEX `FK2_appliance_map` (`appliance_map_id`),"+
      "CONSTRAINT `FK1_owners` FOREIGN KEY (`owner_id`) REFERENCES `owners` (`id`) ON UPDATE CASCADE ON DELETE CASCADE,"
      +
      "CONSTRAINT `FK2_appliance_map` FOREIGN KEY (`appliance_map_id`) REFERENCES `appliance_map` (`id`) ON UPDATE CASCADE ON DELETE CASCADE"
      + ") COLLATE='utf8_general_ci' AUTO_INCREMENT=38")
    int createTestTable();

    @SqlUpdate("drop table appliances")
    int dropTestTable();

}
```

```
@Override
public String toString() {

    return "Device [id=" + id + ", device_id=" + device_id + ", type=" + type + ", user_defined_name="
            + user_defined_name + ", uuid=" + uuid + ", last_upload=" + last_upload + ", last_event_fetch="
            + last_event_fetch + ", owner=" + owner_id + ", is_NILM=" + is_NILM + ", is_registered=" + is_registered
            + ", uuid_bidgely=" + uuid_bidgely + ", capacity=" + capacity + ", direction=" + direction + ", angle="
            + angle + ", withBattery=" + withBattery + ", withLoadManagement=" + withLoadManagement
            + ", installationDate=" + installationDate + ", meta="+ getMeta() + "]";

}
```

… with a „Microservice Architecture“ looking like that !



Constraints of E.ON
(in regards to Microservice considerations)

Market View

- Commodity energy prices for power and gas are under high pressure
- Need to develop innovative products leads to a lot of agile product development MVPs in different domains, like:
  - PV & Battery
  - E-Mobility
  - IoT/ Connected Home
  - Smart Metering

Organizational View

- Old view: Traditional enterprise company with IT as cost-driver
- New view: Product centric organization with IT at its heart
- This includes a number of challenges:
  - Many externals (incl. high fluctuation)
  - Cost-pressure on product dev.
  - Multiple disconnected initiatives
  - No central architecture governance

Motivation of „Microservice Blueprint“

Trigger 1

Problems in existing projects in regards to technical quality and maintainability

Trigger 2

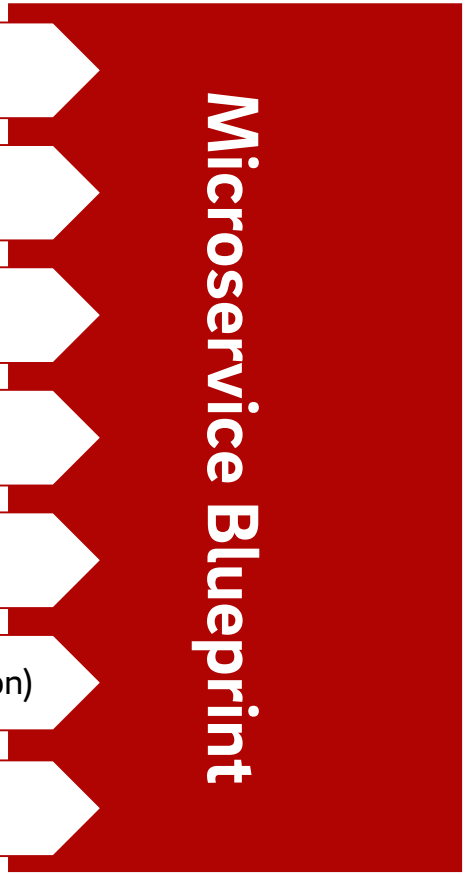
Many starting projects have the same technical requirements of Microservices

Having a **well documented** and a **working reference implementation** for a **java-based Microservice/API development project** taking into account the **best practices** of the last projects

Details of „Microservice Blueprint“

General Remarks in regards to „Microservice Blueprint“

1. Microservice in terms of API not SCS (Self-Contained System)
2. Focus: Java using Spring Boot (+ Docker)
3. Blueprint can be used for “monoliths” as well
4. Should be used as reference not as a mandatory rule
5. Sometimes options are described (e.g. property management)
6. Straightforward example only (relational DB Access + API communication)
7. No common libraries but instead “copy with pride”



Topics covered by „Microservice Blueprint“

Confluence Documentation

General Development Environment

- Setup of IDE
- Setup Postman
- Naming Conventions
- GIT Naming & Branching Conventions
- Versioning and Releasing Conventions
- Setup Development Stage
- CI/CD Setup
- SonarQube Usage
- Splunk Monitoring & Questionnaire
- Docker Deployment

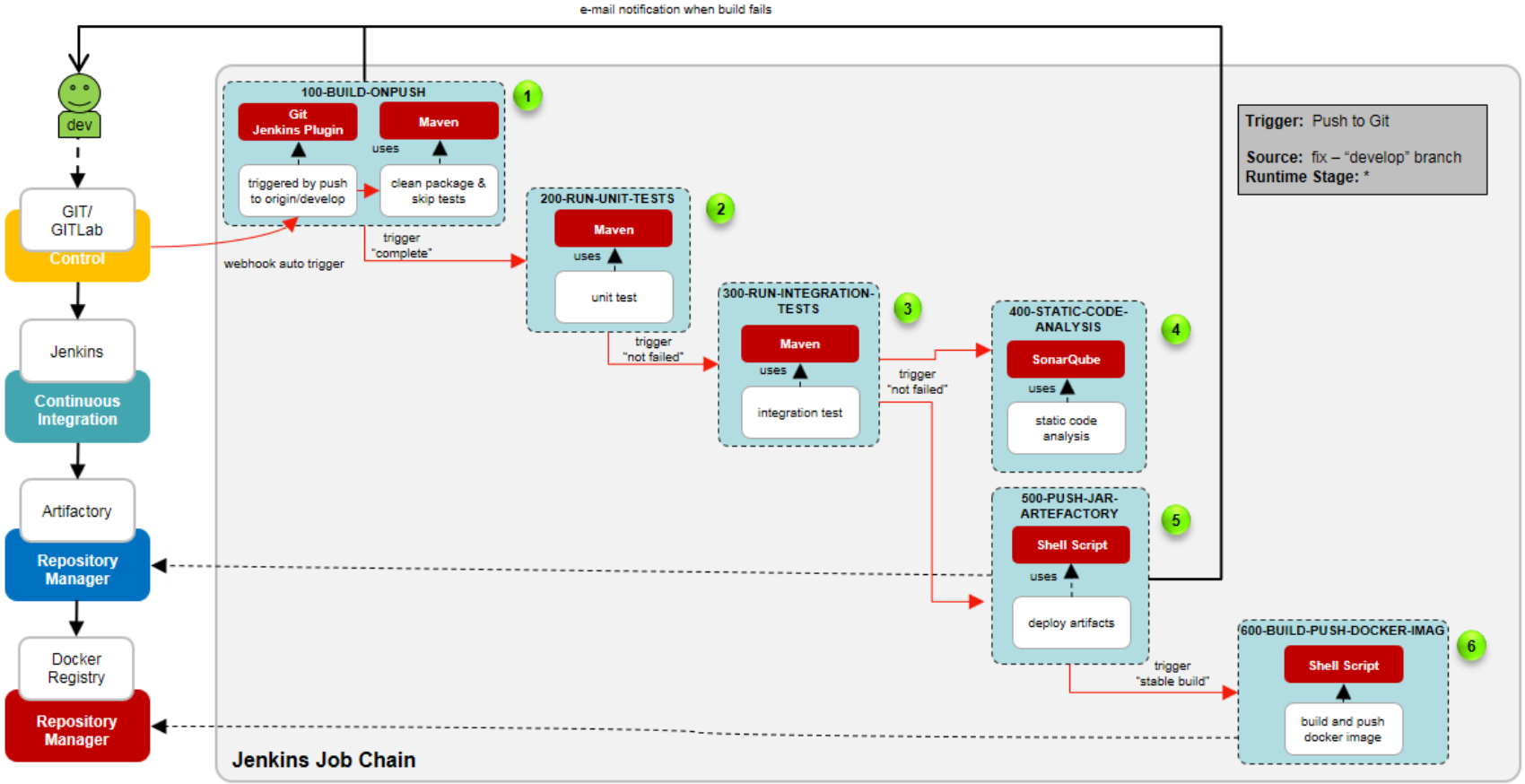
External Facing Considerations

- API Design & Principles
- API Documentation
- Exception Handling for Clients

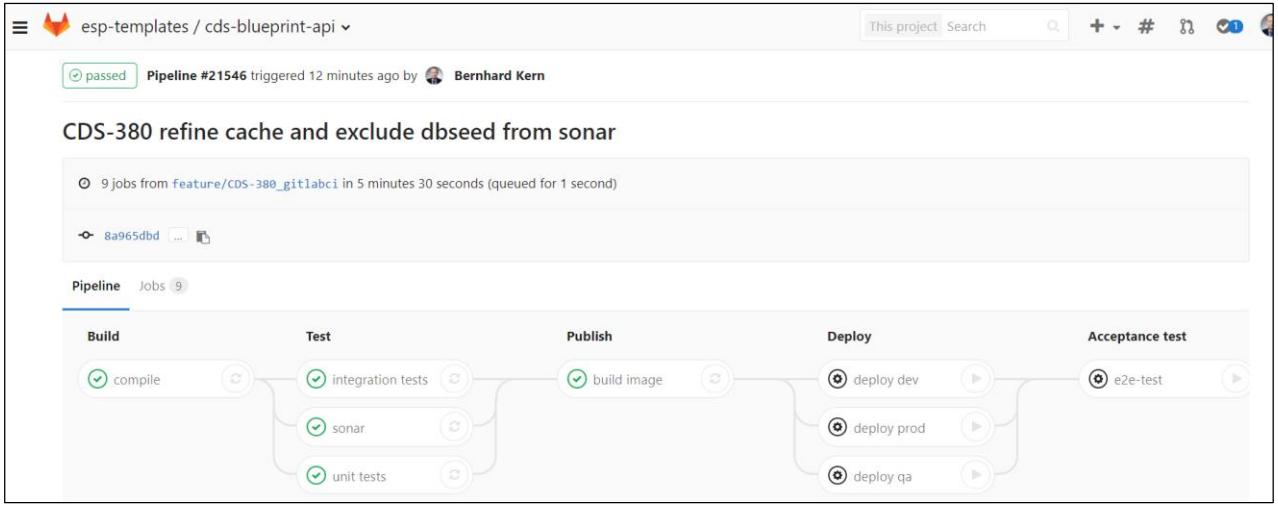
Internal Facing Considerations

- Layering & Project Structure
- Profile & Property Management
- Internal Exception Handling
- Validation (JSR 303)
- Database Access (Spring Data)
- Logging Policy & Format
- Security / API Protection
- Admin Endpoints
- Testing Concept
- Coding Conventions
- REST API Communication
- Usage of HATEOAS

CI/CD - Jenkins



CI/CD – GitLab CI



Exception Handling

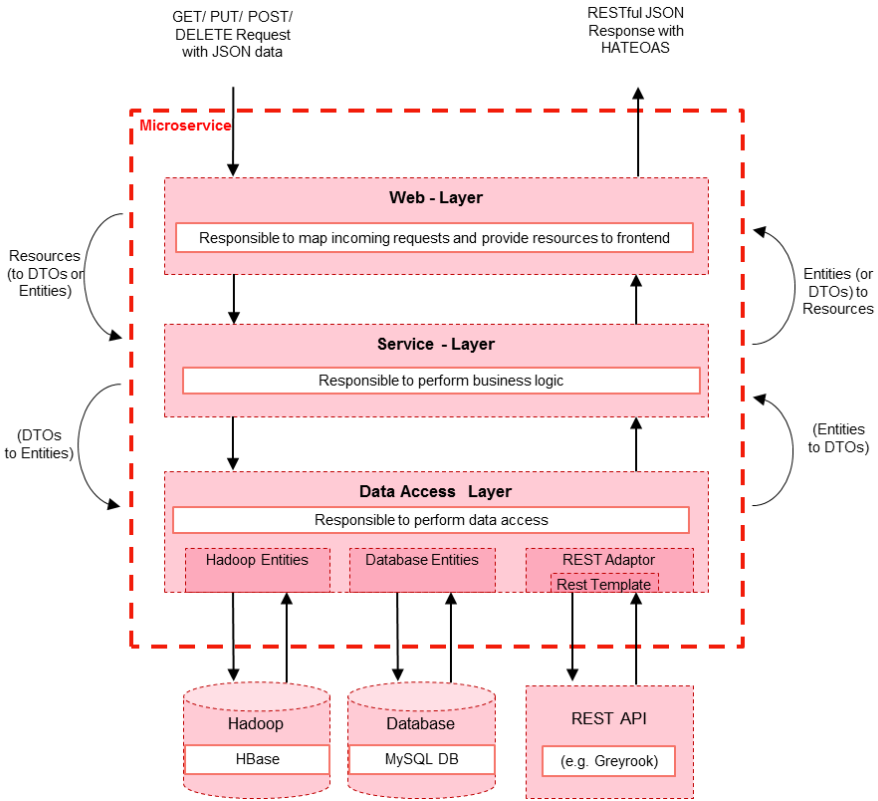
```
{
    "errors": [{
        "code": "ERROR_CODE_FOR_CLIENT_MESSAGE",
        "error-message": "some details"
        "parameters": [{
            "param1",
            "param2"
        }],
        "placeholders": [{
            "name": "the key",
            "type": "the type",
            "value": "the value"
        }]
    }]
}
```

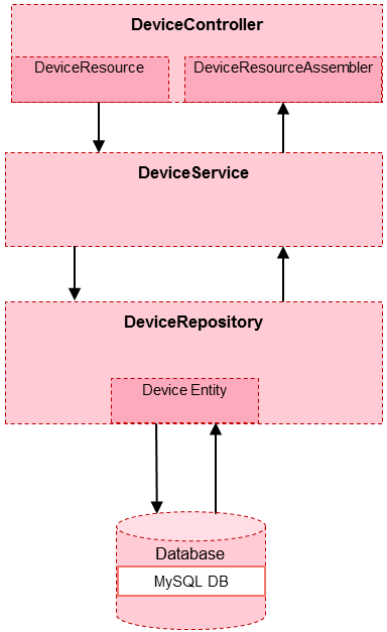
```
{
    "errors": [{
        "code": "ERR_PRODUCT_CHANGE_NOT_ALLOWED",
        "error-message": "Product change not allowed"
        "placeholders": [{
            "name": "allowedProductChangeDate",
            "type": "date",
            "value": "01.05.2018"
        }]
    }]
}
```

Error Conventions:

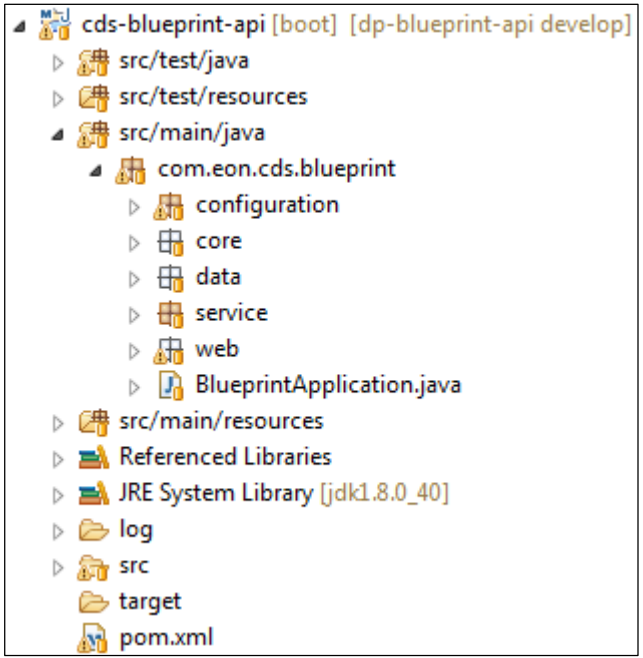
- **4xx** : „You did something wrong“ (e.g. missing parameter, wrong format of parameter, …)
- **5xx** : „We did something wrong“ (e.g. database not available, othere Microservice not reachable, …)
- **Important**: All Microservices/ APIs needs to answer in the same structure!

Layering & Project Structure





Code Walkthrough

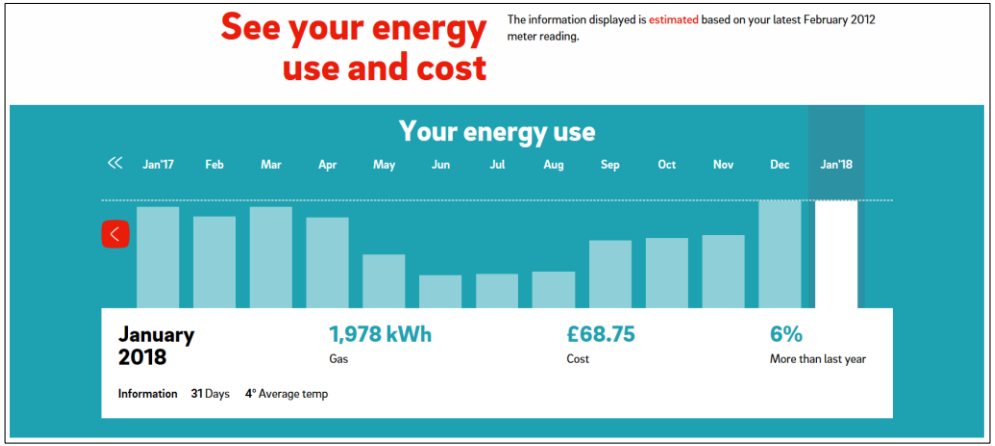


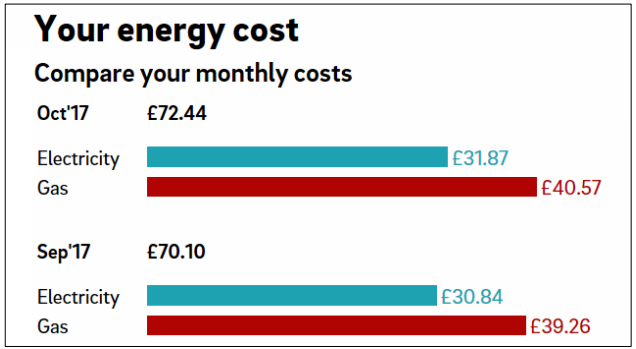
Lets jump into the IDE!

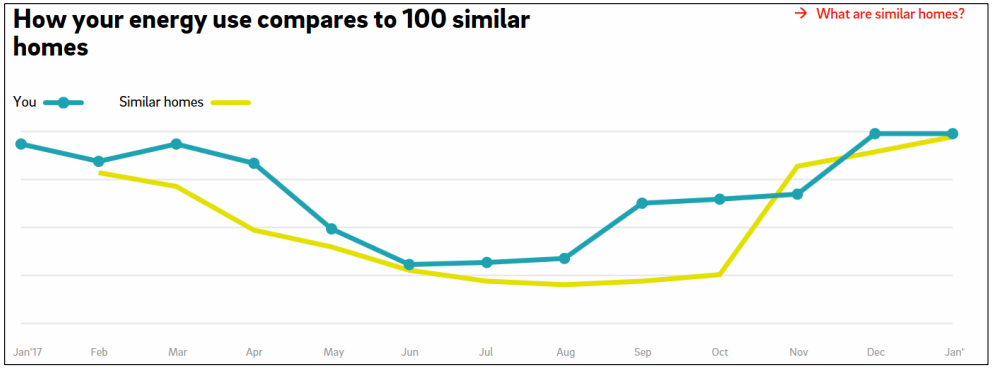
E.ON Saving Energy Tool

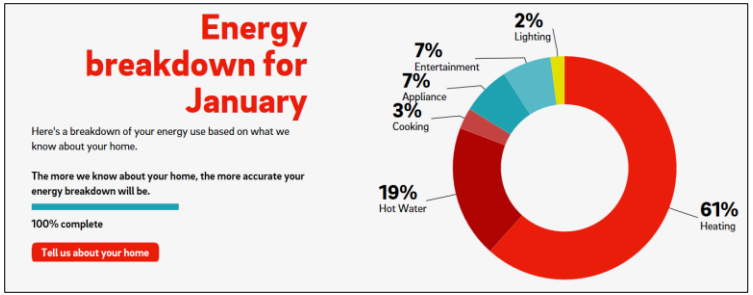
A living example

Functional Overview

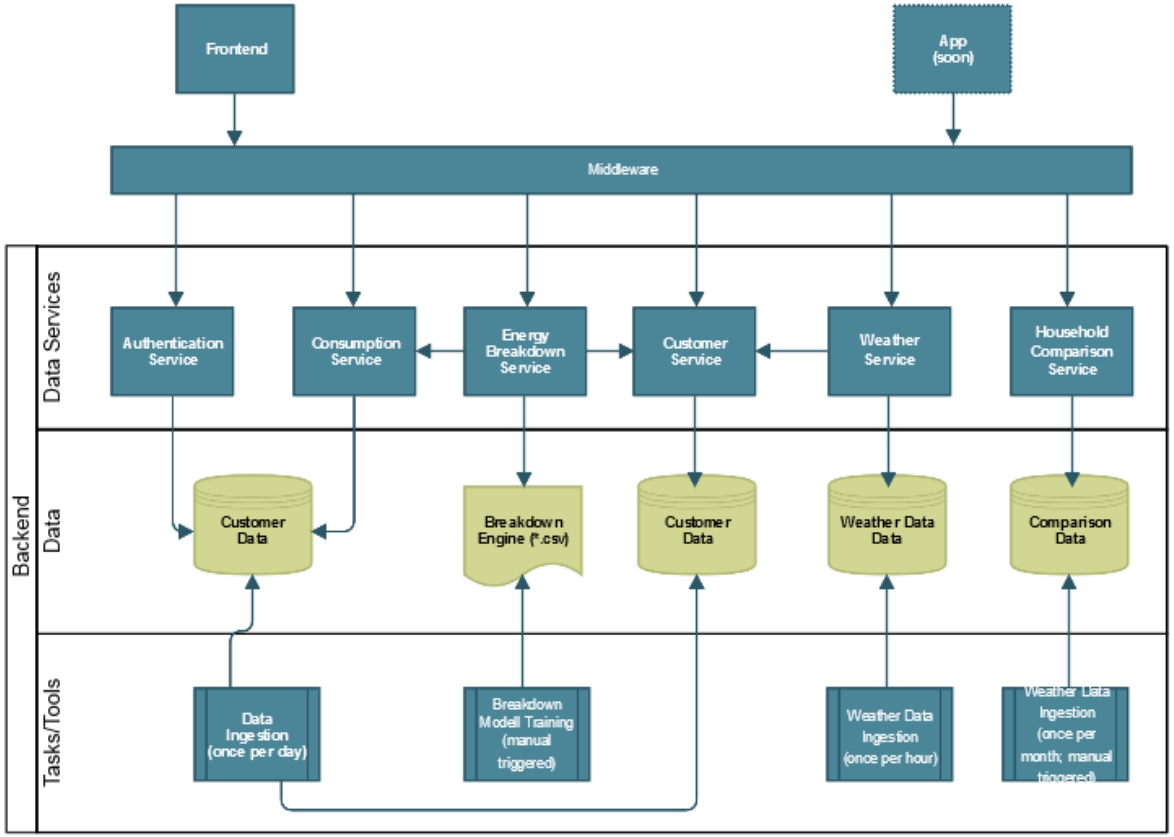






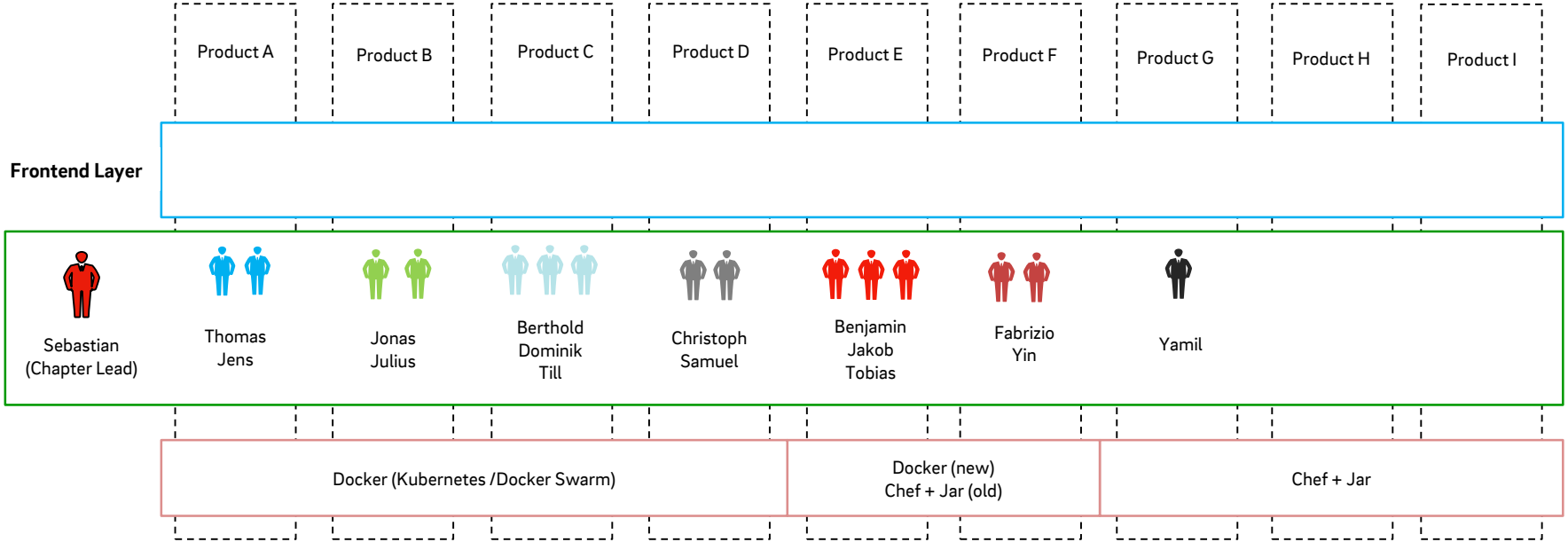


Architecture of Microservices



Data Service/ Middleware API Chapter

Real organization with anonymized names!



Summary

Summary

1. „Microservice Blueprint“ = reference implementation + documentation
2. Best Practices of blueprint has been a proven and used >10 projects
3. First living examples at E.ON available that use microservices
4. Challenge: Multiple small MVP developments rather than one „big“ product
5. Challenge: Stable internal teams (as too many externals)
6. Recommendation: Start with a monolith
7. Recommendation: Chapters are a good way to do knowledge transfer

Microservices @ E.ON

Questions ?

Thank you for your attention!

Sebastian.Eggers@eon.com

BACKUP

Possible Blueprint Setup