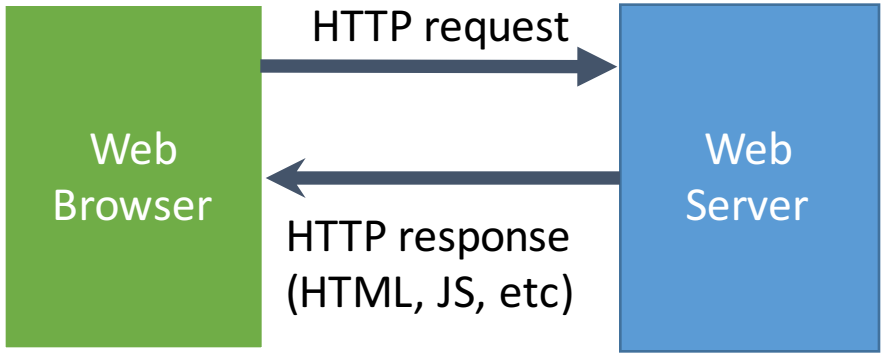
REST APIs, JSON

Niklas Fors, EDAF20, 2019-02-12

HTTP - Hypertext Transfer Protocol



HTTP = Hypertext Transfer Protocol

- transfer protocol between web servers and web browsers, etc.

HTTP supports different types of requests:

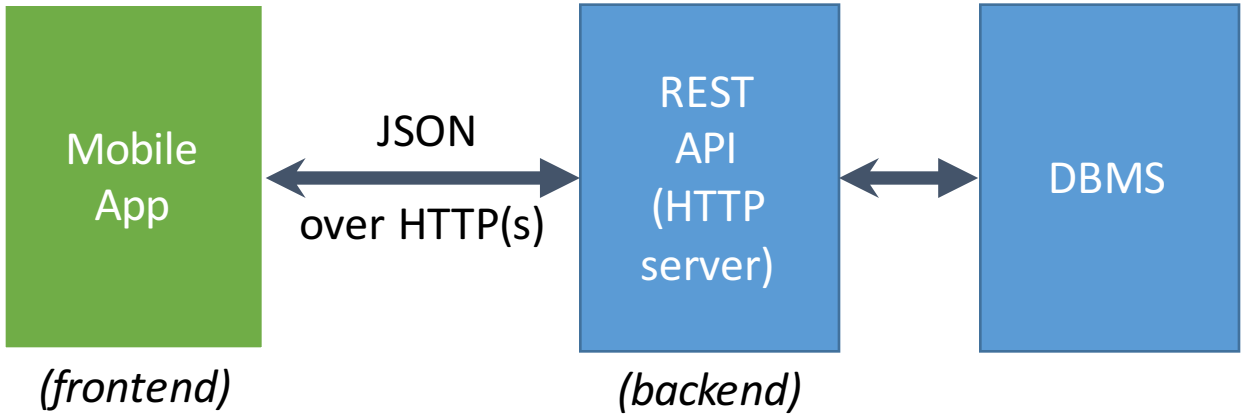
- GET – retrieves data from resource
- POST – creates new resource
- PUT – creates/replaces resource
- PATCH – partial update of resource
- DELETE – deletes resource

HTTP GET Example

curl = library/tool for http, …, requests

```
$ curl http://example.com/index.html
<!doctype html>
<html>
<head>
    <title>Example Domain</title>
    ...
</head>
<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.</p> ...
</div>
</body>
</html>
```

REST – A Common Architecture



JSON = JavaScript Object Notation,

- A common textual encoding standard for transmitting and storing data

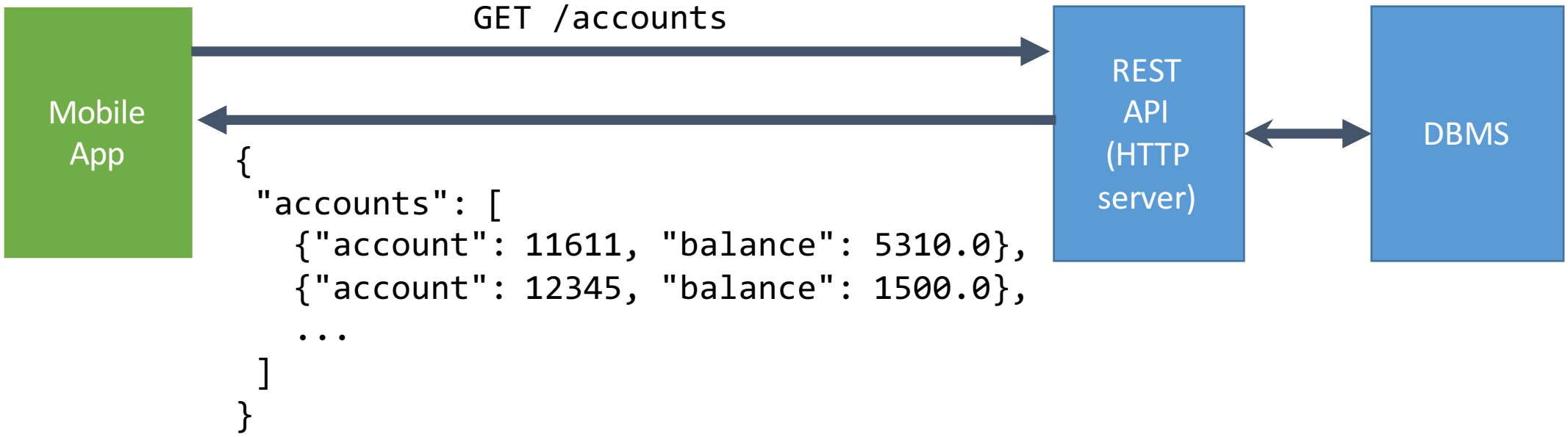
REST = Representational State Transfer

- Used for web services, mobile apps, etc,

API = Application Programming Interface

- Interface of methods/procedures

JSON as response



REST (Representational State Transfer)

- A REST server/service lets clients access and manipulate web resources using stateless operations
- A REST server typically uses HTTP (GET/POST/PUT, …) to communicate with its clients
- Examples of resources
  - `/customers` – all customer
  - `/customers/1` – customer 1

REST Services using HTTP

- To retrieve information, the client makes a GET request to the resource URL
- To create new resources, a POST request is used instead
- Examples:
  - GET /customers/ - get all customers
  - POST /customers/ - creates a new customer
- REST services typically use JSON for data representation

JSON (JavaScript Object Notation)

Supported data types:

- number
- string
- boolean
- array
  - denoted using []
  - ordered list of values
- object
  - denoted using {}
  - unordered collection of name-value pairs
- null

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100”
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234”
    }, {
      "type": "office",
      "number": "646 555-4567”
    }
  ],
  "children": [],
  "spouse": null
}
```

REST GET example with JSON response

```
$ curl -X GET https://jsonplaceholder.typicode.com/todos
[
  {
    "userId": 1,
    "id": 1,
    "title": "delectus aut autem",
    "completed": false
  },
  {
    "userId": 1,
    "id": 2,
    "title": "quis ut nam facilis et officia qui",
    "completed": false
  },
  ...
]
```

Creating a new TODO using POST

```
$ cat new-todo.json
{
  "userId": 1,
  "title": "prepare lecture",
  "completed": false
}

$ curl -X POST -d "@new-todo.json" https://jsonplaceholder.typicode.com/todos
{
  "{  \"userId\": 1,  \"title\": \"prepare lecture\",  \"completed\": false}": "",
  "id": 201
}
```

More REST Examples

Resource	POST	GET	PUT	DELETE
/customers	Create a new customer	Retrieve all customers	Bulk update of customers	Remove all customers
/customers/1	Error	Retrieve the details for customer 1	Update the details of customer 1 if it exists	Remove customer 1
/customers/1/orders	Create new order for customer 1	Retrieve all orders for customer 1	Bulk update of orders for customer 1	Remove all orders for customer 1

Example from https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

HTTP Query Strings

It’s possible to use *query strings* to pass information:

- GET /customers**?country=sweden&firstname=Niklas**
  - Retrieve all customers from Sweden with firstname Niklas
- GET /customers/1/orders**?created-after=2018-01-01**
  - Retrieve all orders for customer 1 created after 2018-01-01

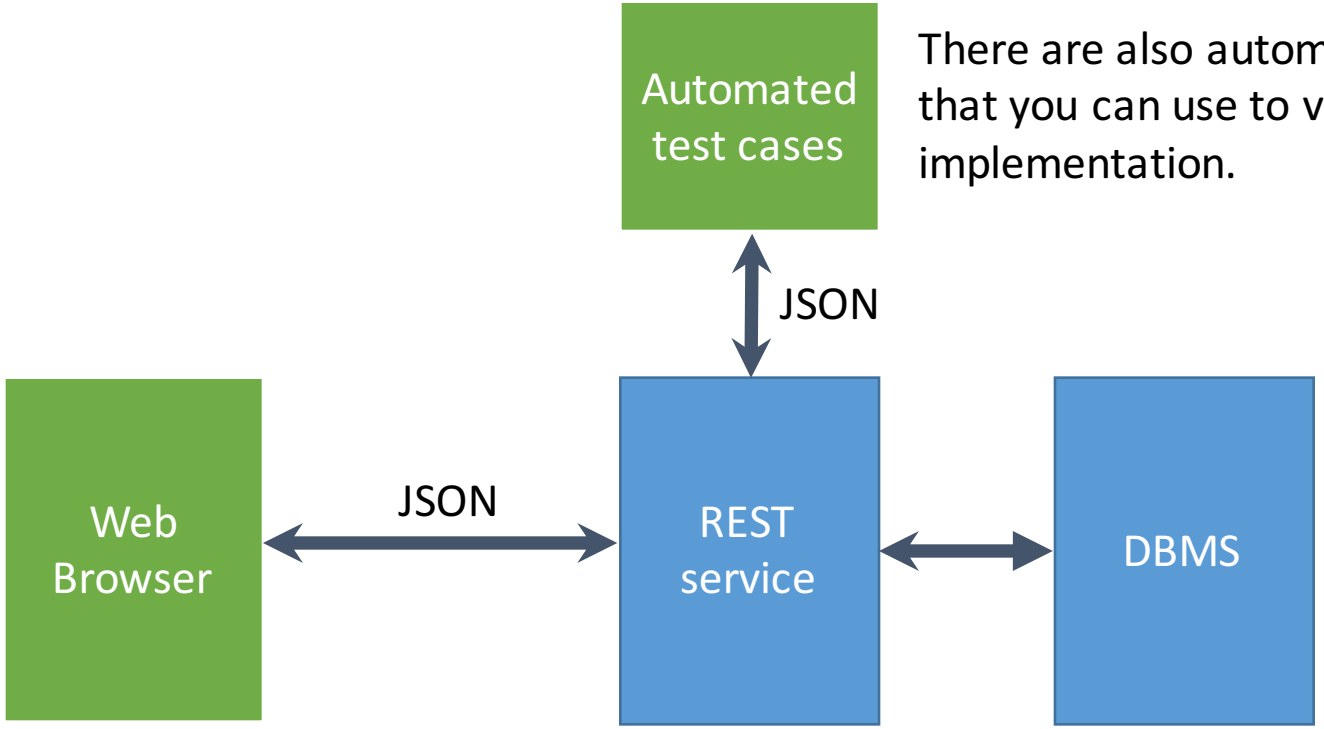
Course Project - Krusty

- Database for cookie production
- The project is divided into several steps:
  1. Design – create E/R model from problem description (natural language)
  2. Peer review of design (exercise session 6 in week 7)
  3. Convert E/R model to relations
  4. Implement relations in DBMS (SQL)
  5. Implement REST API in Java using Spark (skeleton provided)

Course Project Architecture

You will get a web site implemented using React that uses your REST service to display and change data.

First, the web browser retrieves the structure of the page specified in HTML and JavaScript. Then, the JavaScript loads the actual content using the REST service and displays it in the browser.



There are also automated tests that you can use to verify your implementation.

Krusty

Status:

PRODUCTION

Select Cookie

PRODUCE PALLET!

PRODUCED PALLETS

Select Cookie | From åååå-mm-dd | To åååå-mm-dd | Blocked? | FILTER | CLEAR

Name	Production date	Customer	Blocked
Amneris	2019-02-11 20:23:49		no
Amneris	2019-02-11 20:23:49		no
Amneris	2019-02-11 20:23:49		no
Berliner	2019-02-11 20:23:49		no
Nut ring	2019-02-11 20:23:49		no
Nut ring	2019-02-11 20:23:49		no
Tango	2019-02-11 20:23:49		no

RESET DATABASE