

DevOps

DevOpsDemo – die Beispiel-Applikation



Building Competence. Crossing Borders.

REST-Services

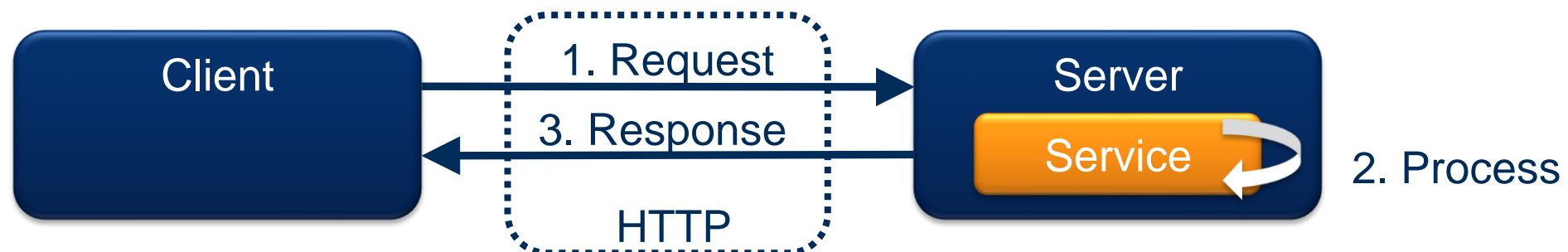
- Eigenschaften von Web-Apps
- Client-Server Architektur
- HTTP-Protokoll
- Routing

Eigenschaften von Web-Apps

Web-Apps sind Anwendungen, die über den Browser aufgerufen werden und keine vorgängige Installation erfordern.



Web-Apps basieren auf der Client-Server Architektur. Die Kommunikation zwischen Client und Server erfolgt hauptsächlich über das HTTP-Protokoll.



HTTP-Anfragemethoden

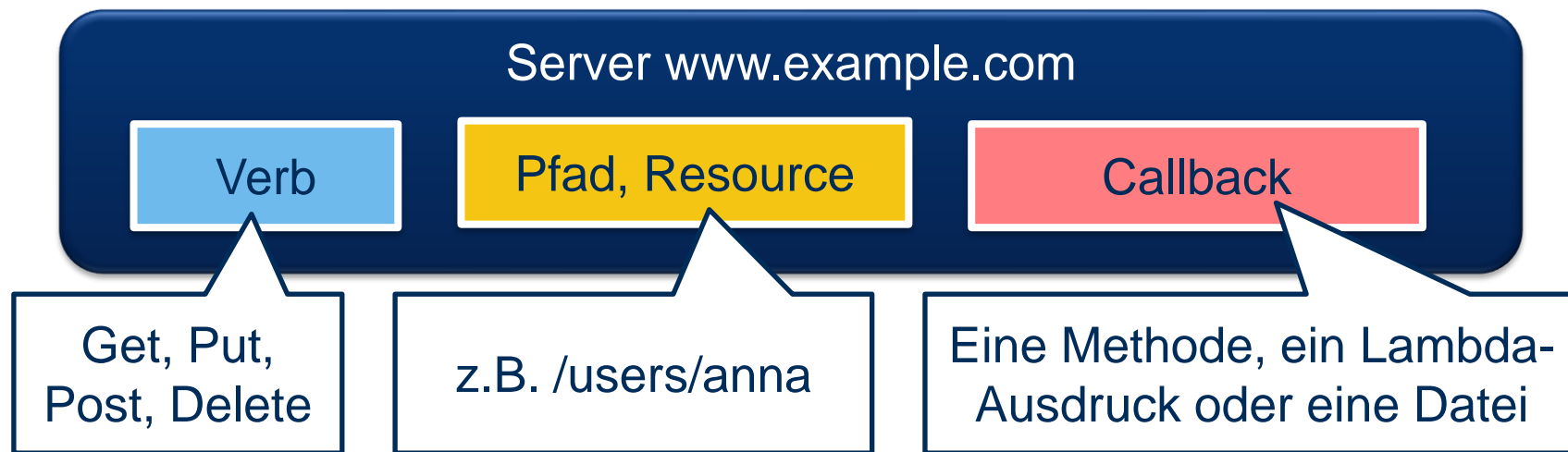
Es gibt verschiedene Typen von Anfragen (Requests). Die wichtigsten sind:

GET	Fordert eine bestimmte Ressource (Webseite, Datei) vom Server an. Wird im Browser die URL www.example.com/users/anna eingegeben, erhält der Server example.com eine GET-Anfrage für die Ressource /users/anna .
POST	Schickt Daten, beispielsweise von Formularen, an den Server. Erzeugt ein neues Objekt auf dem Server.
PUT	Übermittelt eine Ressource (z.B. eine Datei) an den Server. Ändert ein bestehendes Objekt falls dieses bereits existiert.
DELETE	Löscht die angegebene Ressource auf dem Server

Get-Request können mit dem Web-Browser gemacht werden, indem die entsprechende URL eingegeben wird. Für die anderen Methoden braucht es JavaScript Code auf der Webseite. Mit Entwicklungswerkzeugen wie z.B. Postman können alle Methoden benutzt werden.

Routing und Routes

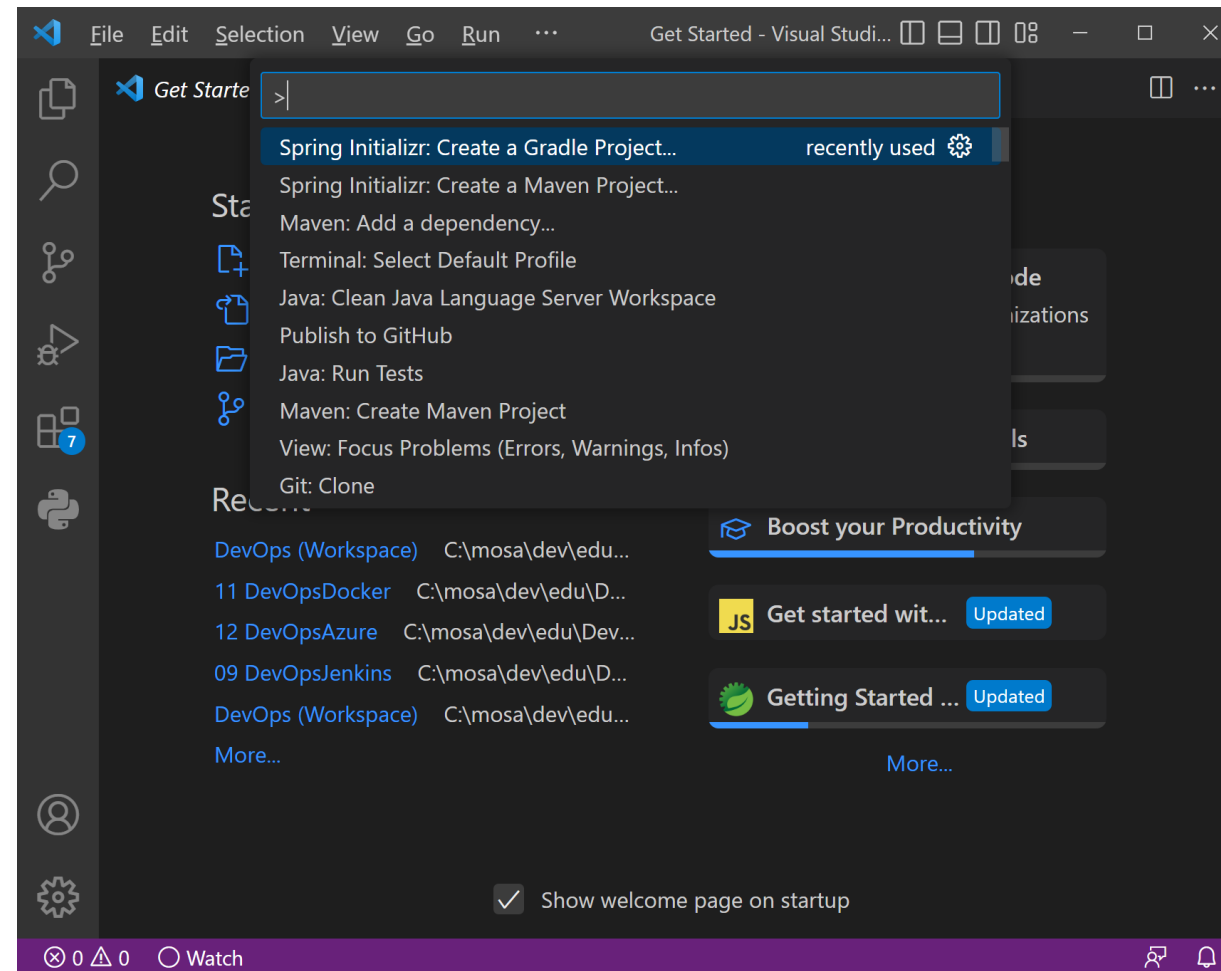
Damit der Webserver weiss, welche Applikation oder Methode für die Behandlung einer Anfrage zuständig ist, muss beim Webserver das **Routing** konfiguriert werden. Dieses besteht aus mehreren Routes. Eine einzelne Route hat den folgenden Aufbau:



Wird im Browser die Seite www.example.com/users/anna aufgerufen, prüft der Server ob es eine passende Route gibt. Wenn ja, wird der definierte Callback ausgeführt.

Spring Boot Hello World

Ein neues SpringBoot-Projekt kann mit Spring Initializr erstellt werden:

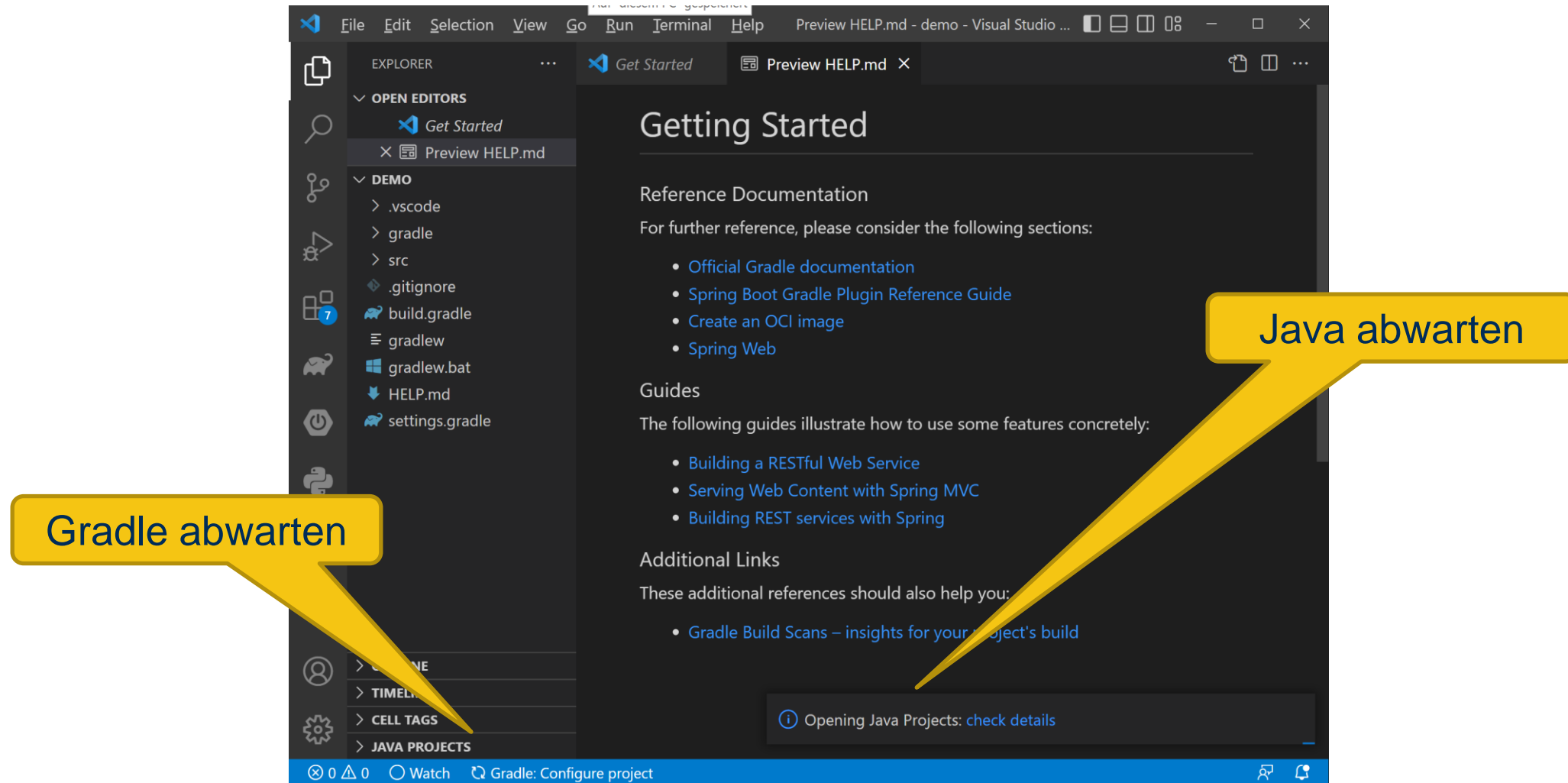


Mit dieser Anleitung lässt sich ein neues SpringBoot-Projekt erstellen:

1. Visual Studio Code starten
2. Ctrl-Shift-P: Spring Initializr: Create a Gradle Project...
3. Specify Spring Boot Version: 3.2.1 (oder vergleichbar)
4. Specify Project Language: Java
5. Input Group Id: ch.zhaw.springboot
6. Input Artifact Id: demo
7. Specify Packaging Type: Jar
8. Specify Java Version: 21 (oder Ihre bisher verwendete Version)
9. Search for Dependencies: Spring Web
10. Verzeichnis erstellen, Generate into this folder
11. Info-Message in Visual Studio Code: Open
12. Yes, I trust the authors

A solid blue background with the text "Neues Spring Boot-Projekt" in white, bold, sans-serif font on the left. On the right, there is a yellow, tilted rectangular tag with the word "Tutorial" in black, sans-serif font.

A blue gradient background with a yellow label in the top right corner that says "Tutorial".



Projekt starten: einige Möglichkeiten

Tutorial

The image shows a VS Code interface with a project named '04a DevOpsSpringBoot'. The left sidebar displays the 'GRADLE' task list, with 'application' expanded and 'bootRun' selected. A yellow callout bubble with the text 'mit Gradle' points to the 'bootRun' task. The main editor shows the 'DemoApplication.java' file. The bottom panel displays the 'OUTPUT' window with the following log:

```
Start to build: testClasses classes
Task :compileJava started
Task :compileJava SUCCESS
Task :processResources started
Task :processResources SUCCESS
Task :classes started
Task :classes SUCCESS
Task :compileTestJava started
Task :compileTestJava UP-TO-DATE
Task :processTestResources started
Task :processTestResources skipped
Task :testClasses started
Task :testClasses UP-TO-DATE
BUILD SUCCESSFUL
```

A yellow callout bubble with the text 'Menu' points to the 'Run' menu in the top right corner. The menu is open, showing options like 'Start Debugging', 'Run Without Debugging', 'Stop Debugging', etc. Below the VS Code interface, a web browser window is open at 'localhost:8080', displaying a 'Whitelabel Error Page' with the message: 'This application has no explicit mapping for /error, so you are seeing this as a fallback. Tue Jan 10 08:23:46 CET 2023 There was an unexpected error (type=Not Found, status=404).'.

Noch kein REST-Service vorhanden, die «Fehlermeldung» auf Port 8080 ist in Ordnung.

REST-Service mit Spring Boot

Analog DevOpsDemo Backend

RestController

```
package ch.zhaw.springboot.demo;

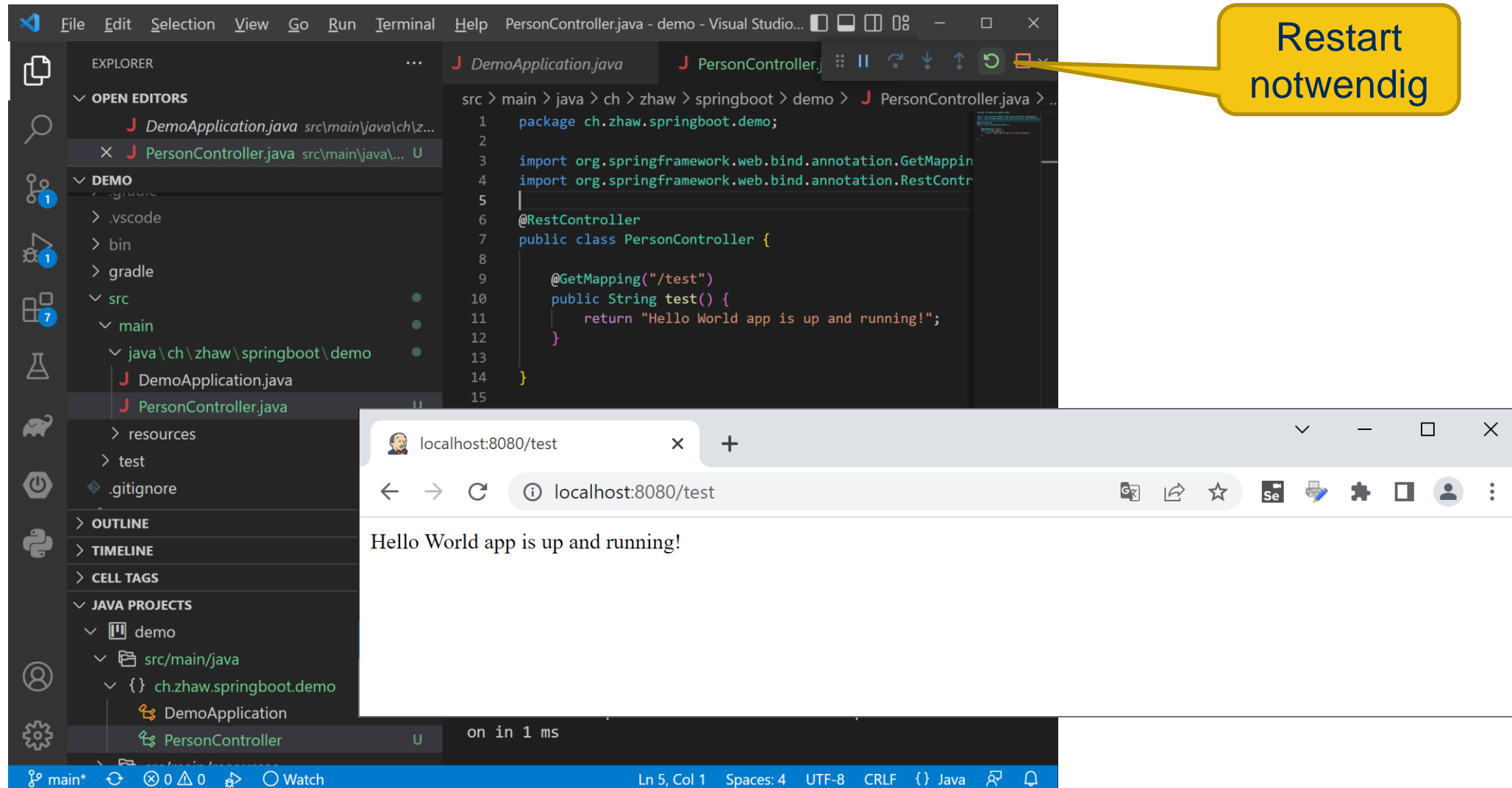
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PersonController {

    @GetMapping("/test")
    public String test() {
        return "Hello World app is up and running!";
    }

}
```

RestController testen



Einfache Personenverwaltung

Eine Klasse «Person» mit id (Integer) und name (String)

```
package ch.zhaw.springboot.demo;  
  
public class Person {  
  
    private int id;  
    private String name;  
  
    public Person() {}  
  
    public Person(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
}
```

Personen anzeigen

Erweiterung PersonController für «GET Person» mit ID

```
@RestController
public class PersonController {

    private Map<Integer, Person> persons = new HashMap<>();

    @EventListener(ApplicationReadyEvent.class)
    public void init() {
        this.persons.put(1, new Person(1, "Barack Obama"));
        this.persons.put(2, new Person(2, "Donald Trump"));
        this.persons.put(3, new Person(3, "Joe Biden"));
        System.out.println("Init Data");
    }

    @GetMapping("/person/{id}")
    public Person getPerson(@PathVariable Integer id) {
        return this.persons.get(id);
    }

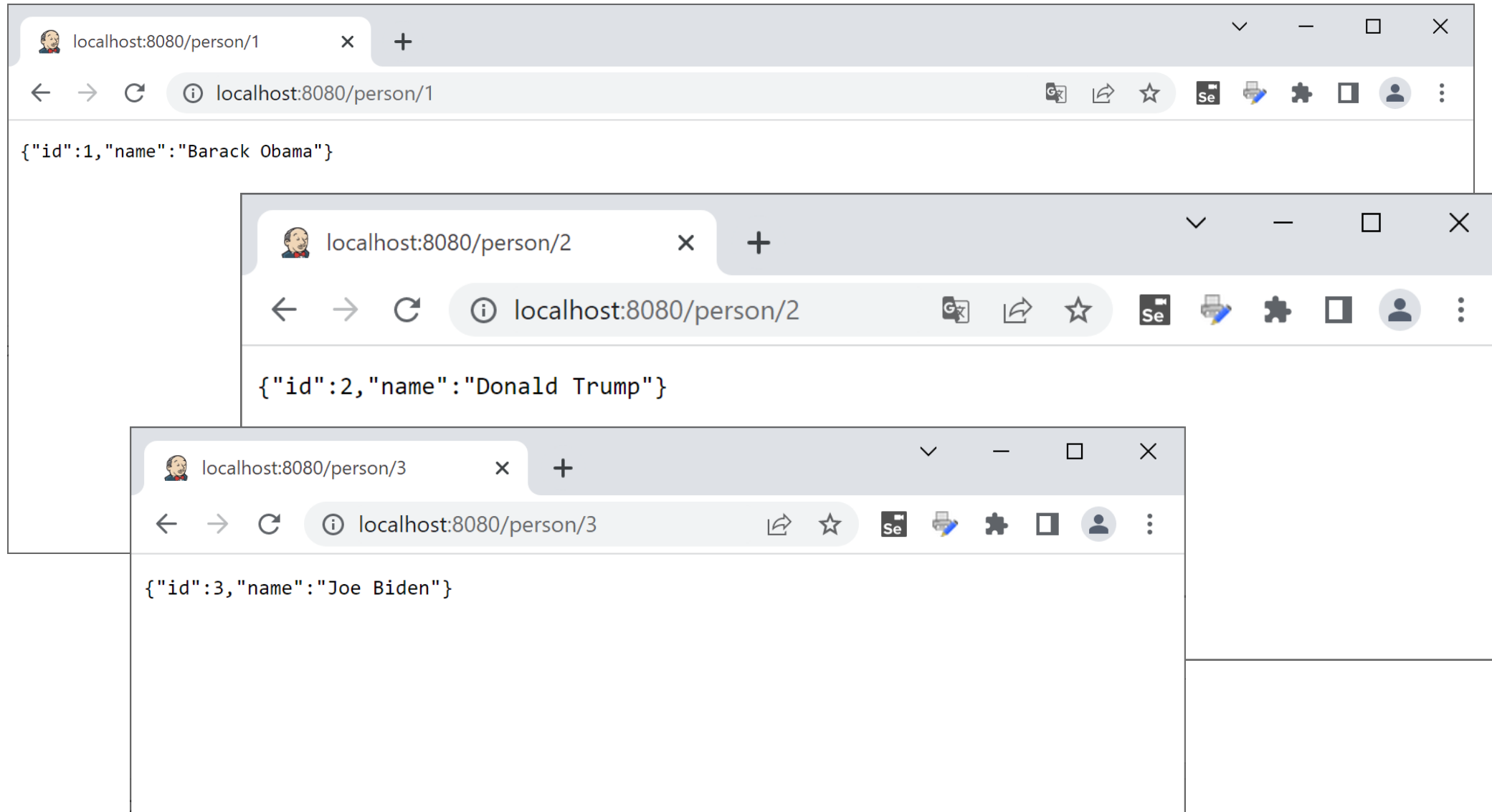
}
```

Map aller
Personen

Methode
wird beim
Start
einmalig
ausgeführt

GET für
Person

GET Person testen



POST, PUT, DELETE

Bei POST wird eine neue, maximale ID erfasst.

```
@PostMapping("/person")
public void createPerson(@RequestBody Person todo) {
    var newId = this.persons.keySet().stream().max(Comparator.naturalOrder()).orElse(0) + 1;
    todo.setId(newId);
    this.persons.put(newId, todo);
}

@PutMapping("/person/{id}")
public void updatePerson(@PathVariable Integer key, @RequestBody Person person) {
    person.setId(key);
    this.persons.put(key, person);
}

@DeleteMapping("/person/{id}")
public Person deletePerson(@PathVariable Integer id) {
    return this.persons.remove(id);
}
```

Aber wie können diese Services getestet werden?

Testen mit Postman

Aufruf des Service mit Postman

Vorbedingungen

- Registration bei <https://www.postman.com/>
- Installation des Postman Desktop Agent

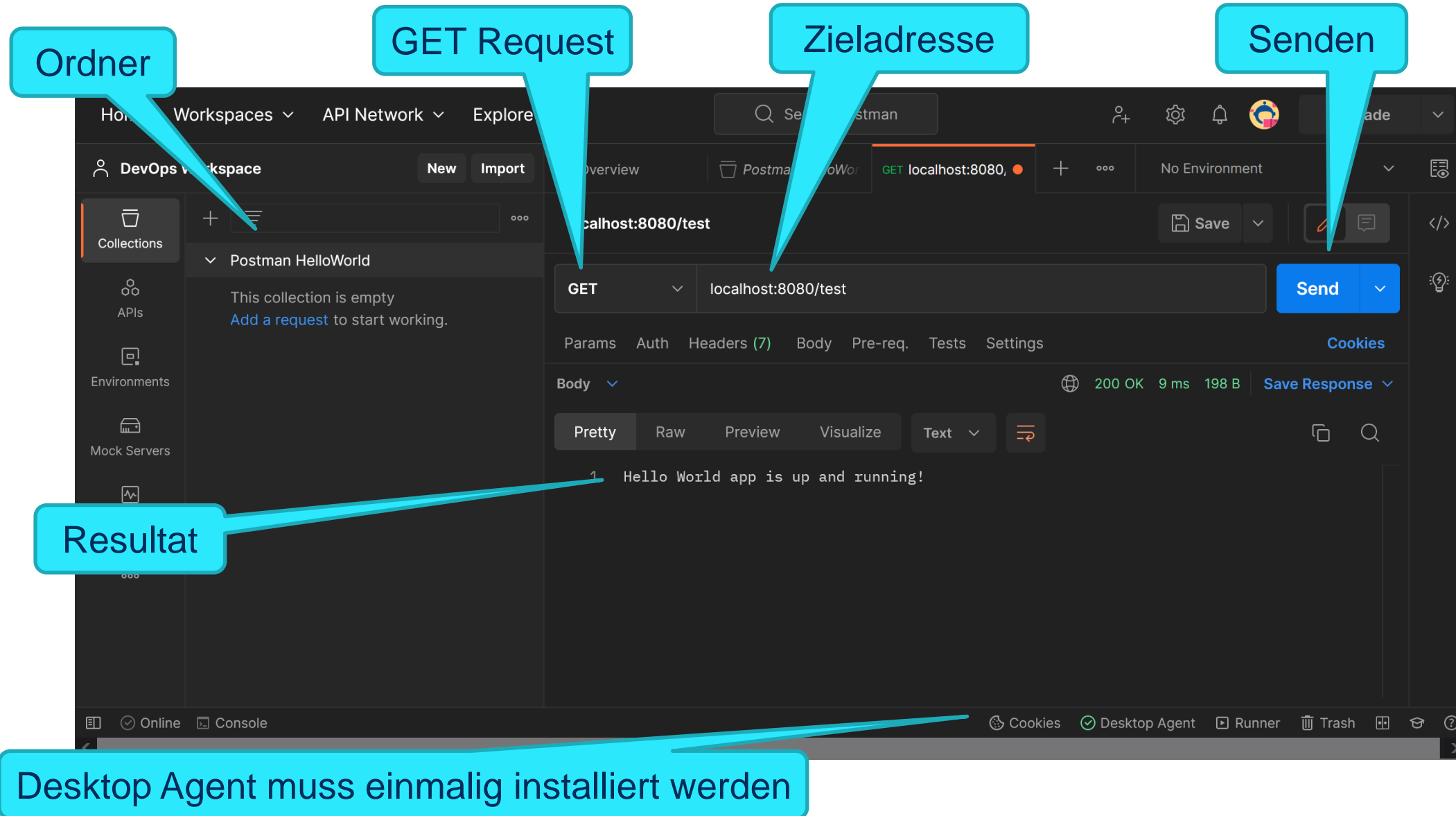
Postman

- Simulation von Backend-Requests
- Im Gegensatz zu Browser-URL-Feld auch komplexe Aufrufe wie POST-Requests

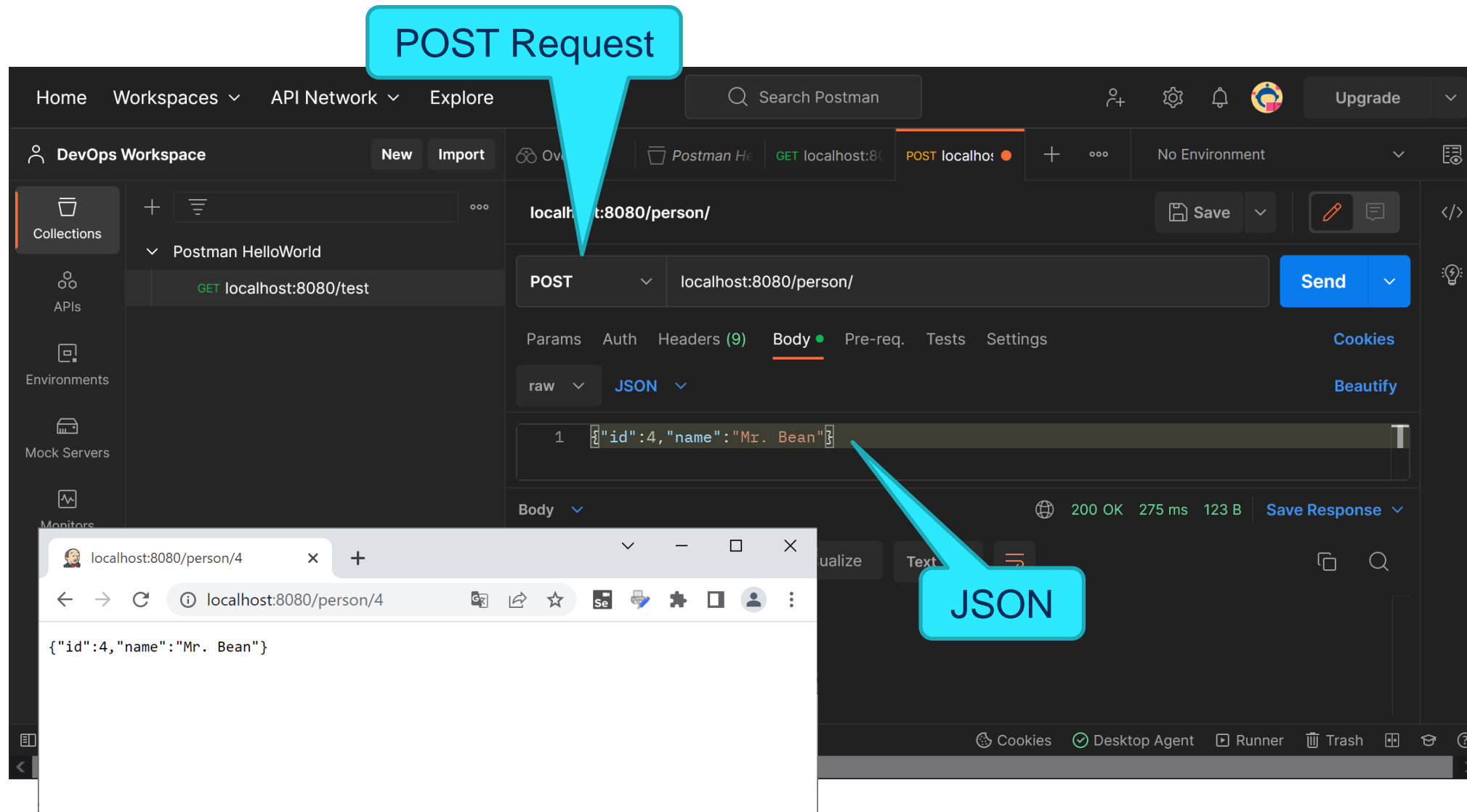
Postman Desktop Agent

- Postman läuft im Browser
- Postman Desktop Agent ermöglicht Zugriff auf localhost

Postman Hello World mit GET-Request



POST Beispiel



DevOpsDemo Frontend

Path als Frontend-Framework

DevOpsDemo – Beispiel-Applikation mit Path und Spring Boot

DevOpsDemo lokal aufsetzen

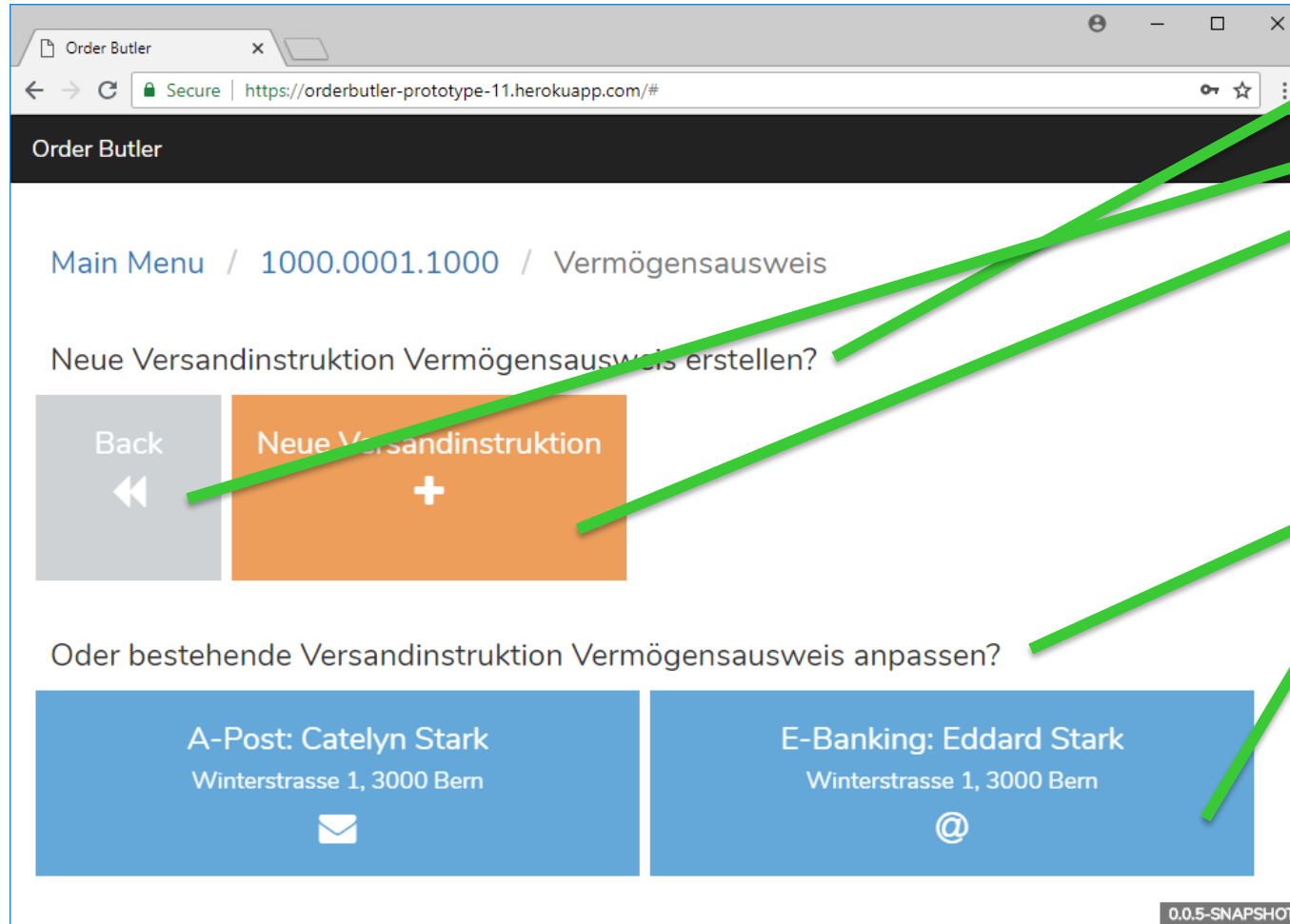
Path Frontend Framework

Ein einfaches Frontend
für unsere
Beispiel-Applikation

Path UI Modell
Path Architecture
Path Apps
Examples
Path Backend



Was ist ein UI Modell?

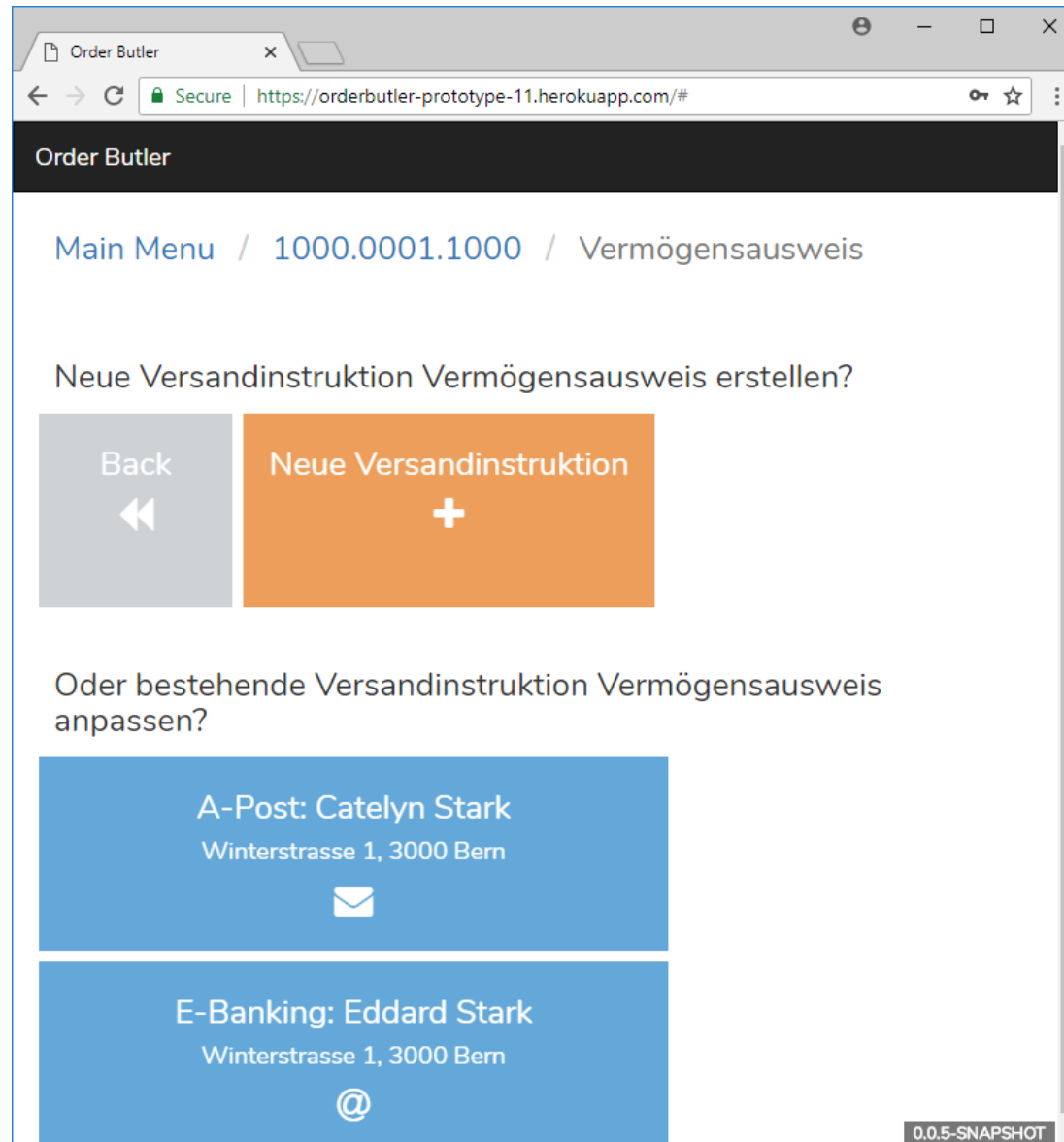


Modell

- **Text-Element**
- **Backbutton**
- **Einzel-Button** mit Text, Icon und Ziel

- **Text-Element**
- **Button-Liste** mit Service-URL (Datenherkunft)

Beispiel für Path UI Modell



```
{
  "id": "orderDepotAssetStatement",
  "elementList": [
    {
      "type": "pageLabel",
      "newRow": true,
      "value": "Neue ... erstellen?"
    },
    {
      "type": "button",
      "name": "NewDepotAssetStatement",
      "icon": "fa-plus",
      "color": "carrot",
      ...
    },
    {
      "type": "pageLabel",
      "newRow": true,
      "value": "Oder bestehende ... anpassen?"
    },
    {
      "type": "list",
      "icon": "fa-building",
      "color": "green-sea",
      "width": 3,
      "search": false,
      "page": "...",
      "url": "/orderDepotAssetStatement/...",
    },
  ],
}
```

Path Forms

Eingabefelder (Text, Datum, Zahl, ...) sowie CRUD-Operationen werden unterstützt. Auch die Forms sind als UI-Modell abgelegt. Daten werden als JSON übermittelt.

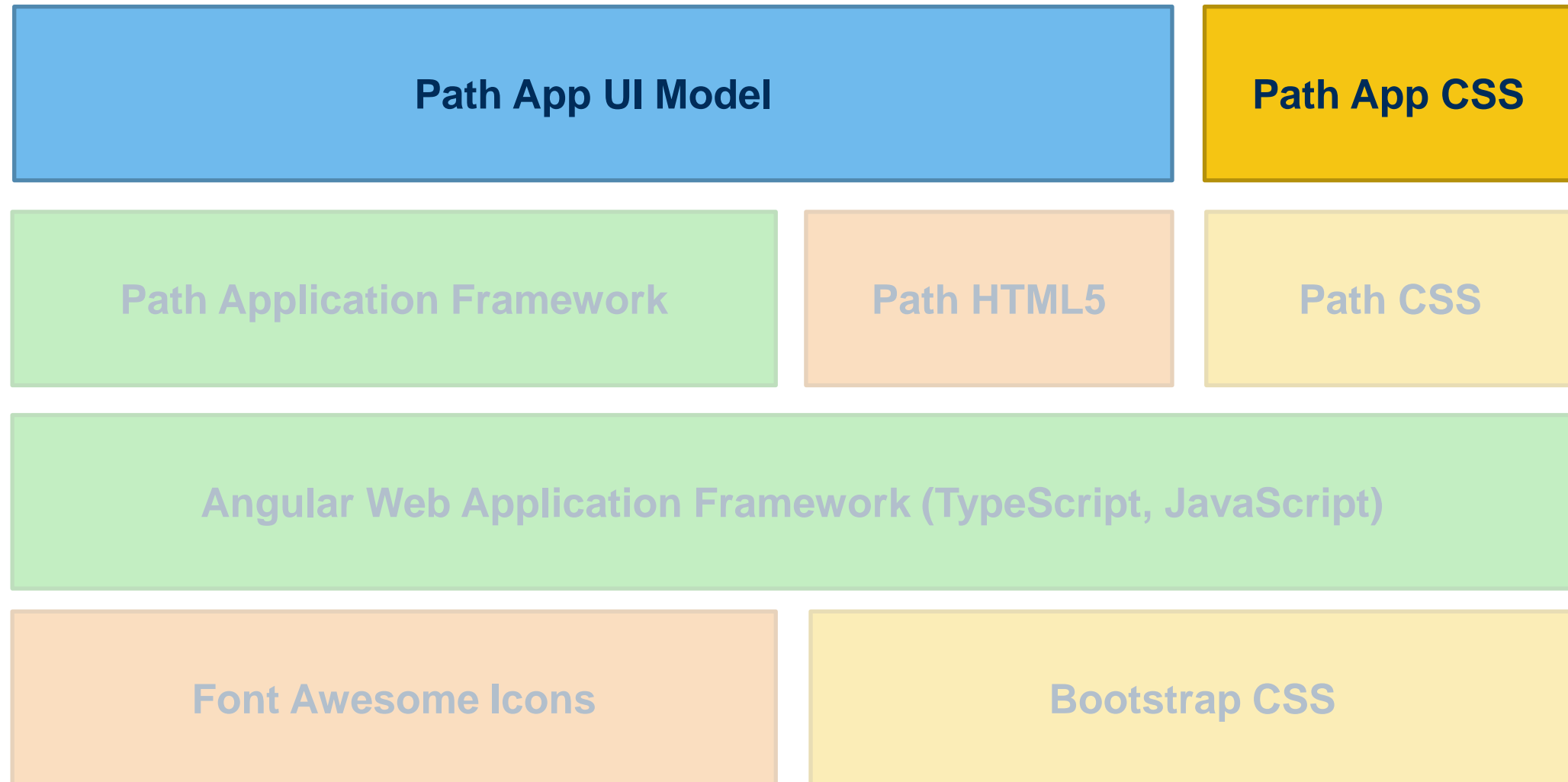
The screenshot shows a web browser window with the URL `https://path-example.herokuapp.com`. The application is titled "Path Example App" and shows a user is signed in as "demo". A modal window titled "Project" is open, displaying a form for creating or editing a project. The form fields are:

- Project Name:** Text input field containing "Project A".
- Company:** Text input field containing "ZHAW" with a "Detail..." link.
- Start date:** Date picker showing "09/04/2018".
- End date:** Date picker showing "09/10/2018".
- Priority:** Radio buttons for "Low", "Medium", and "High" (selected).
- Type:** Checkboxes for "Product" and "Service" (checked).
- Comments:** Text area.

At the bottom of the modal are buttons for "Delete", "Cancel", and "Ok". The version number "0.3.8" is visible in the bottom right corner of the application window.

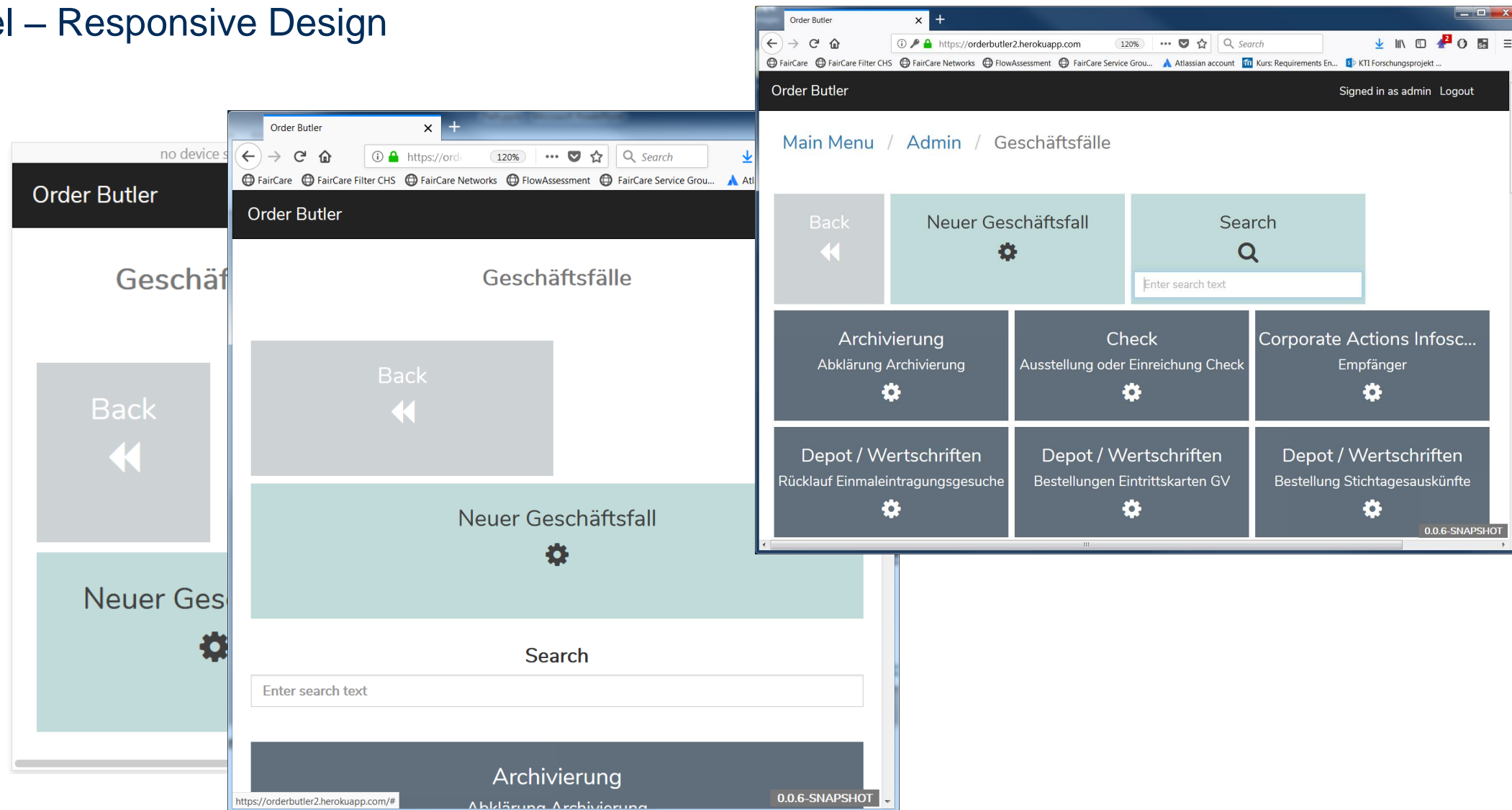
```
{  
  "name": "Project A",  
  "company": "company_d43c8222",  
  "evtStart": "2018-09-04T00:00:00.000Z",  
  "evtEnd": "2018-09-10T00:00:00.000Z",  
  "projectPriority": "2",  
  "projectType": [  
    "Service"  
  ]  
}
```

Path Frontend Architecture

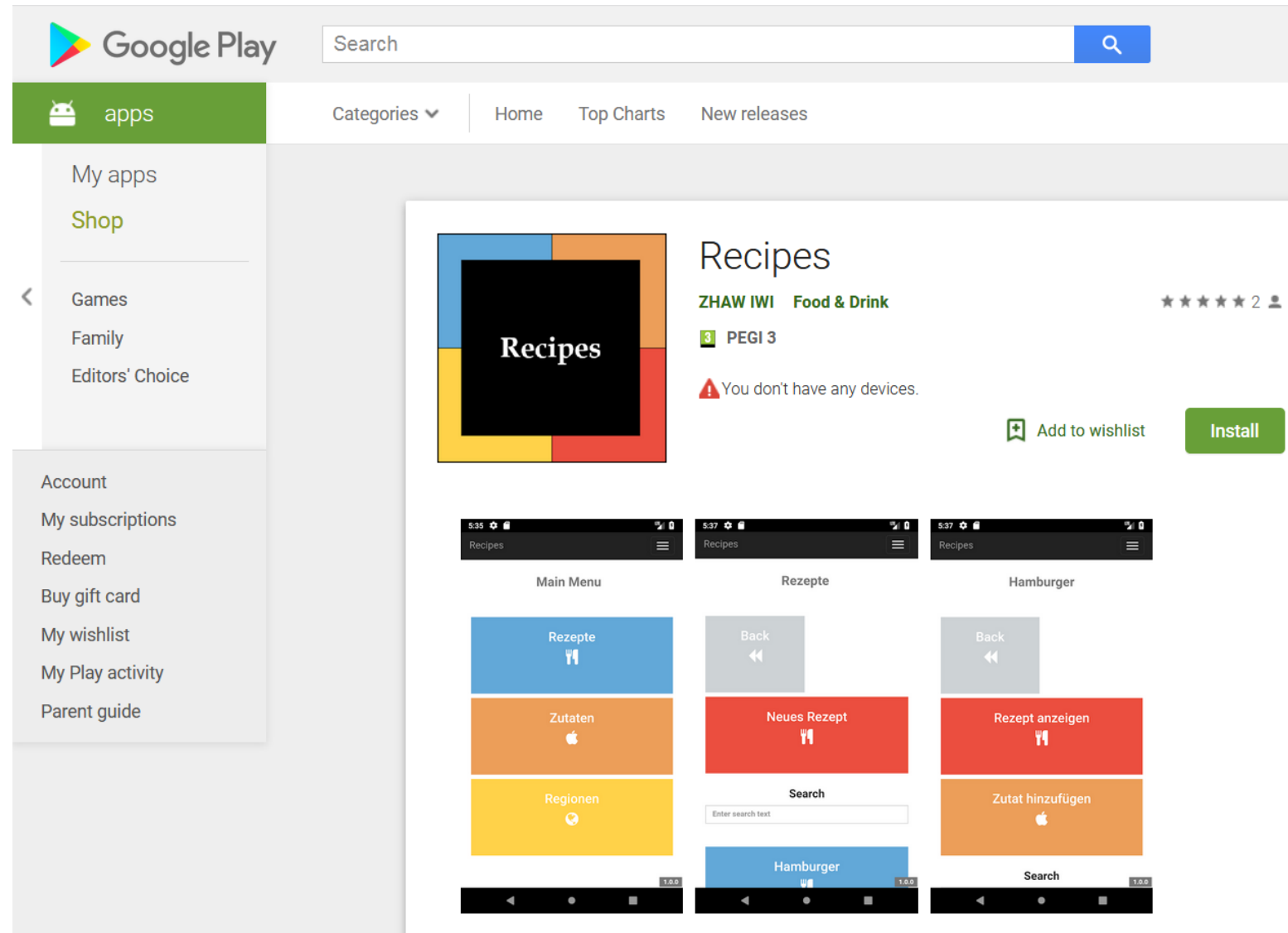


Mobile, Tablet, Desktop

Same Model – Responsive Design



Hybride Path Apps mit Apache Cordova



Path UI / UX Customizing

Path Application

- Die Path App benötigt keinen HTML5 Code, besitzt nur ein UI-Modell
- Das UI / UX kann angepasst werden
 - Farben / Schriftarten / etc. über das Path App CSS
 - Im UI Modell enthaltene Optionen können gesteuert werden
 - Für weitere Anpassungen muss das UI Modell von Path erweitert werden

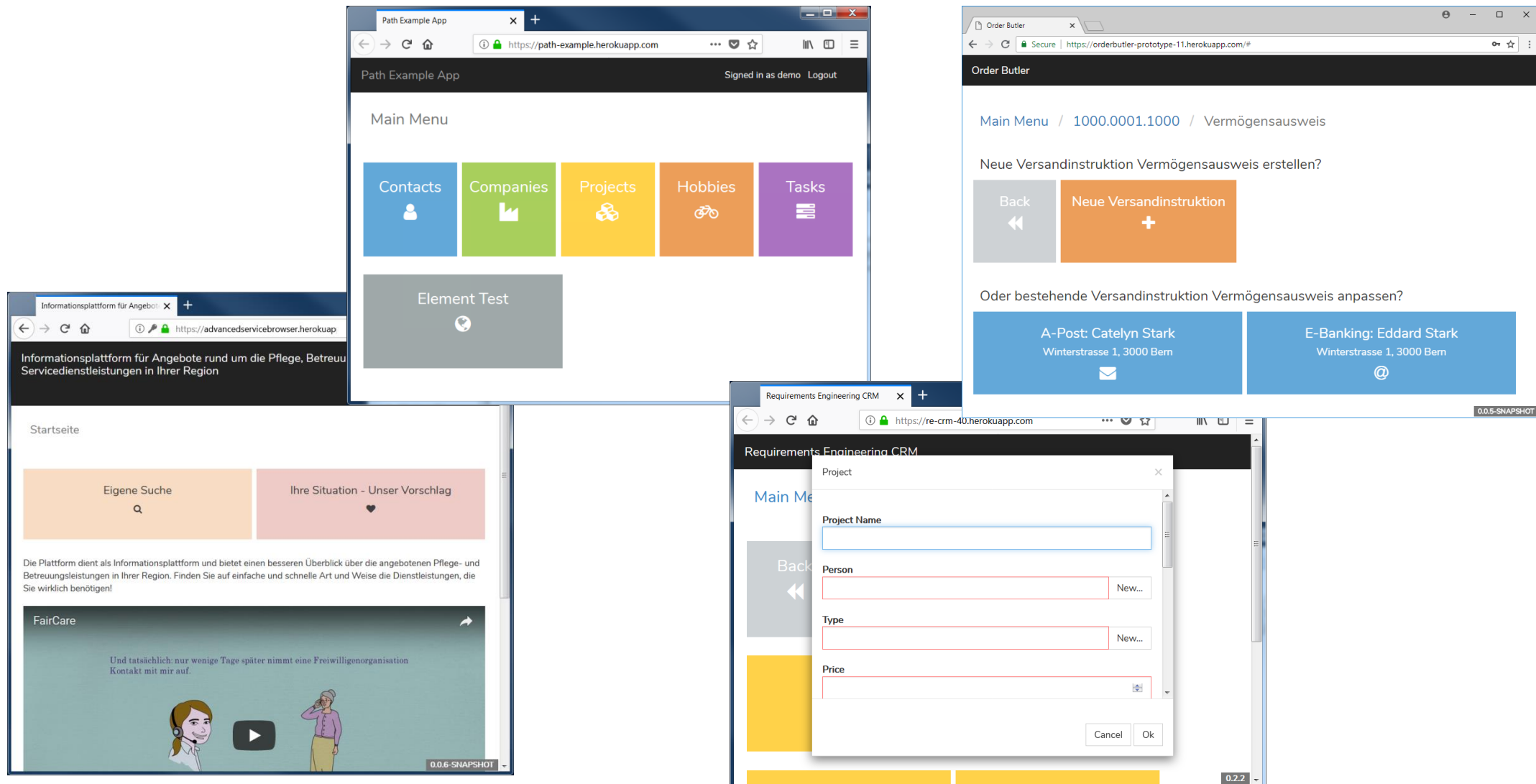
Path Framework (Open-Source, ZHAW)

- Definiert alle zur Verfügung stehenden Optionen für das UI Modell
- Rendert das UI Modell des OrderButlers mit Hilfe von Angular, Bootstrap und Font Awesome

Bootstrap / Font Awesome Icons

- Standard-Libraries für Webdesign und Icons

Path Web Apps: Beispiele



Path Backend

Jedes Backend mit REST-Services wird unterstützt. Beispielimplementierungen:

Java

Java

Hibernate (Persistence)

Spring Boot

Spring Boot Web (REST)

Gradle (Build System)

Postgres / H2 (Database)

Node

Javascript / Typescript

NoSQL Database

Express (REST)

NPM (Node Package Manager)

CouchDB / PouchDB

DevOpsDemo aufsetzen: Tutorial

DevOpsDemo-Applikation auf dem eigenen Rechner installieren

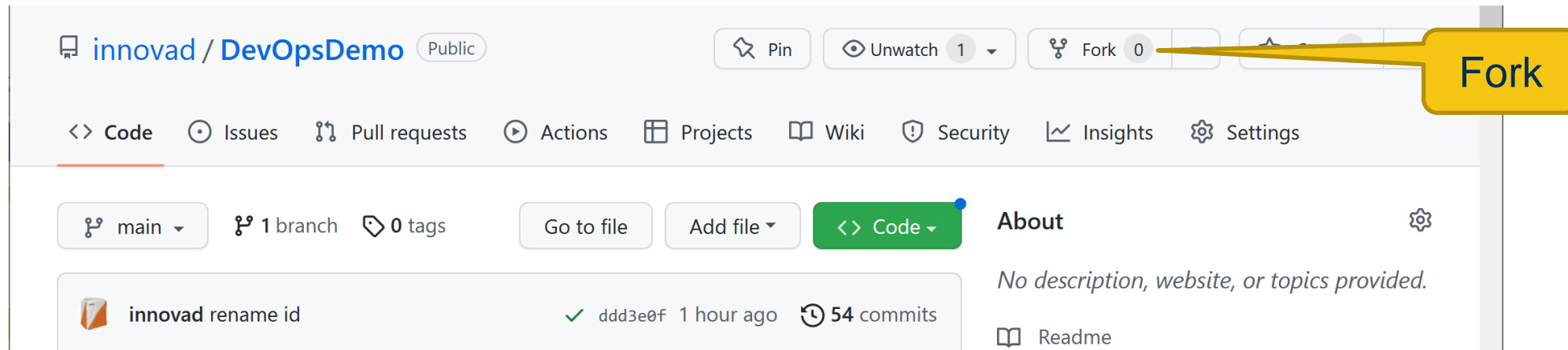
Im späteren Verlauf der Vorlesung werden wir DevOpsDemo weiterverwenden

DevOpsDemo

- <https://github.com/mosazhaw/DevOpsDemo>
- Projekt forken, danach **neue** URL verwenden!

Fork

Eine Kopie auf GitHub. Im Gegensatz zu einem Klon wird die Kopie vom Original getrennt, so dass diese bearbeitet und angepasst werden kann

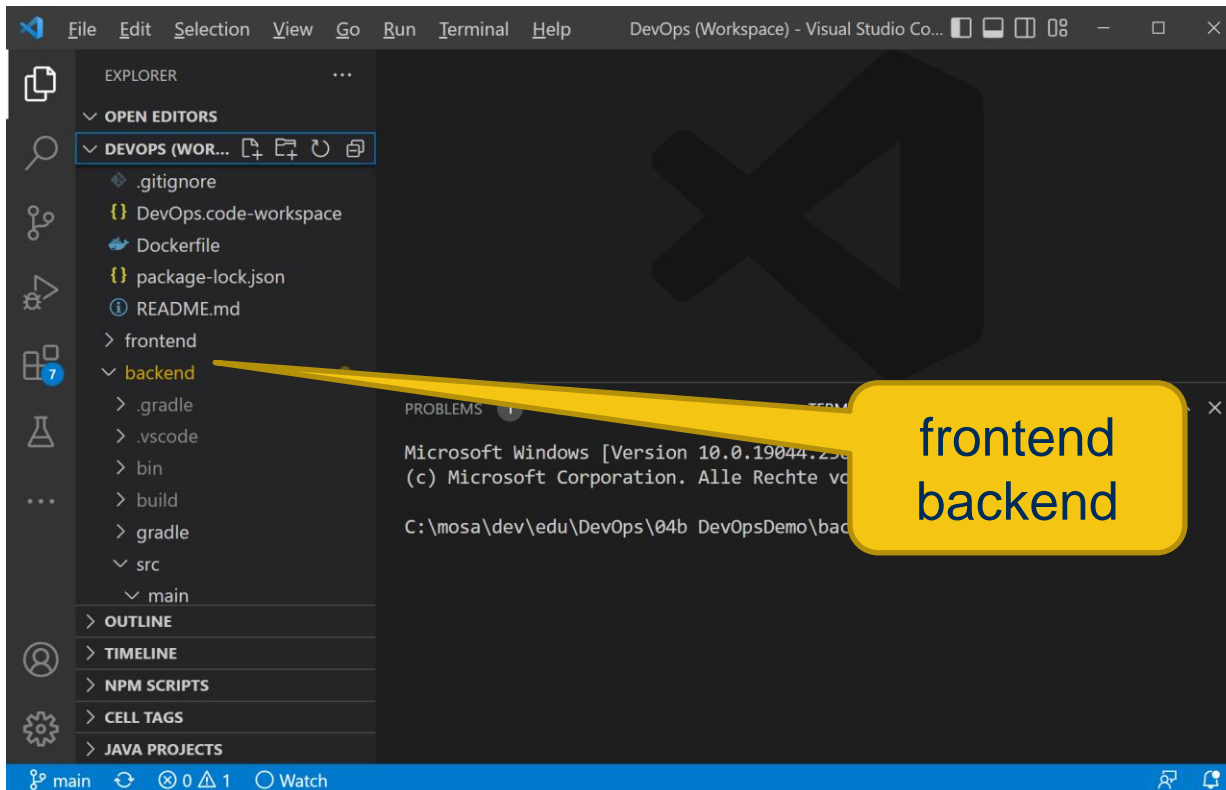


In Visual Studio Code öffnen (File → Open Workspace...)

Unterordner

- frontend: mit NPM/HTML/Javascript gebaut
- backend: mit Gradle/Java gebaut

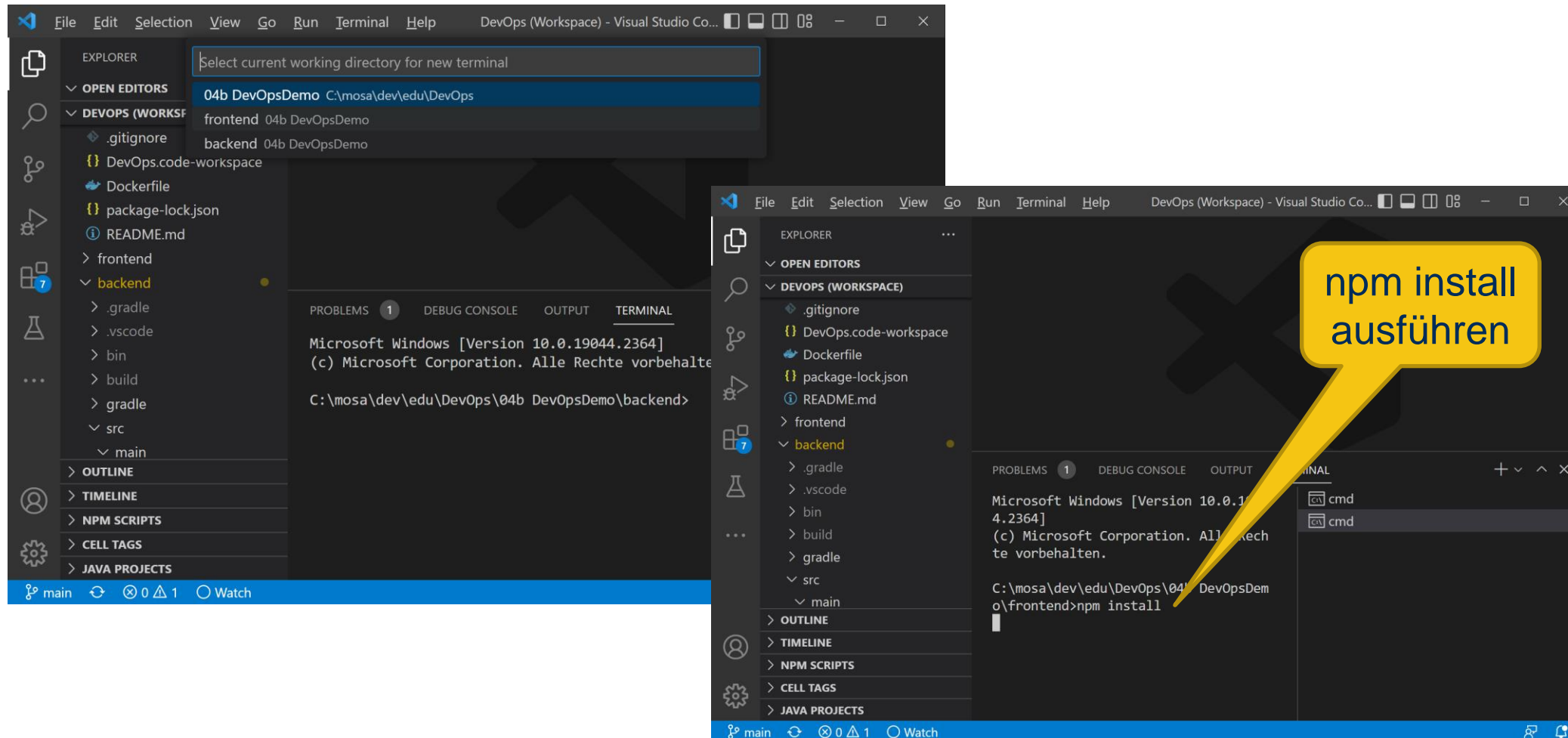
Diese Projekt hat eine Workspace-Definition hinterlegt. Diese **muss** geöffnet werden.



Vorbereitung: NPM Dependencies laden

Tutorial

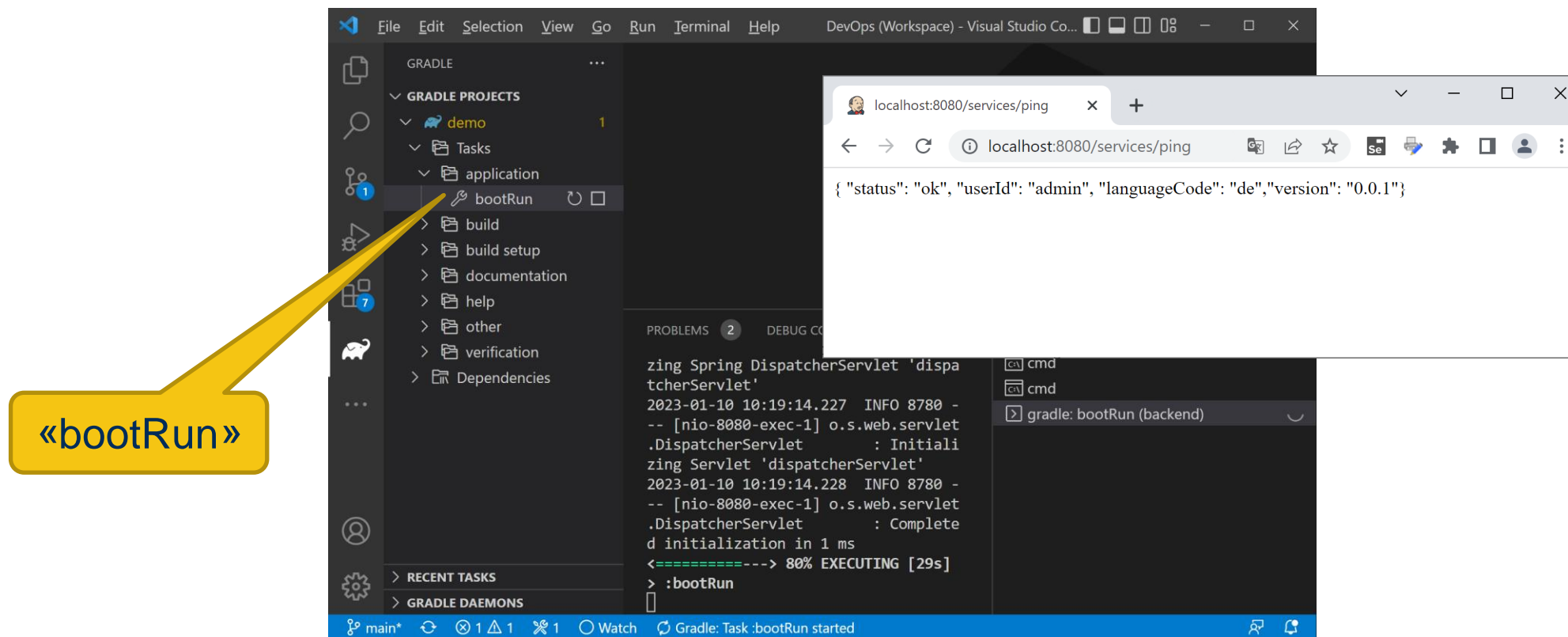
Konsole/Terminal in VS Code für **frontend** öffnen:



Backend starten und prüfen (mit Gradle)

Tutorial

Das Backend kann mit **Gradle** gestartet werden:



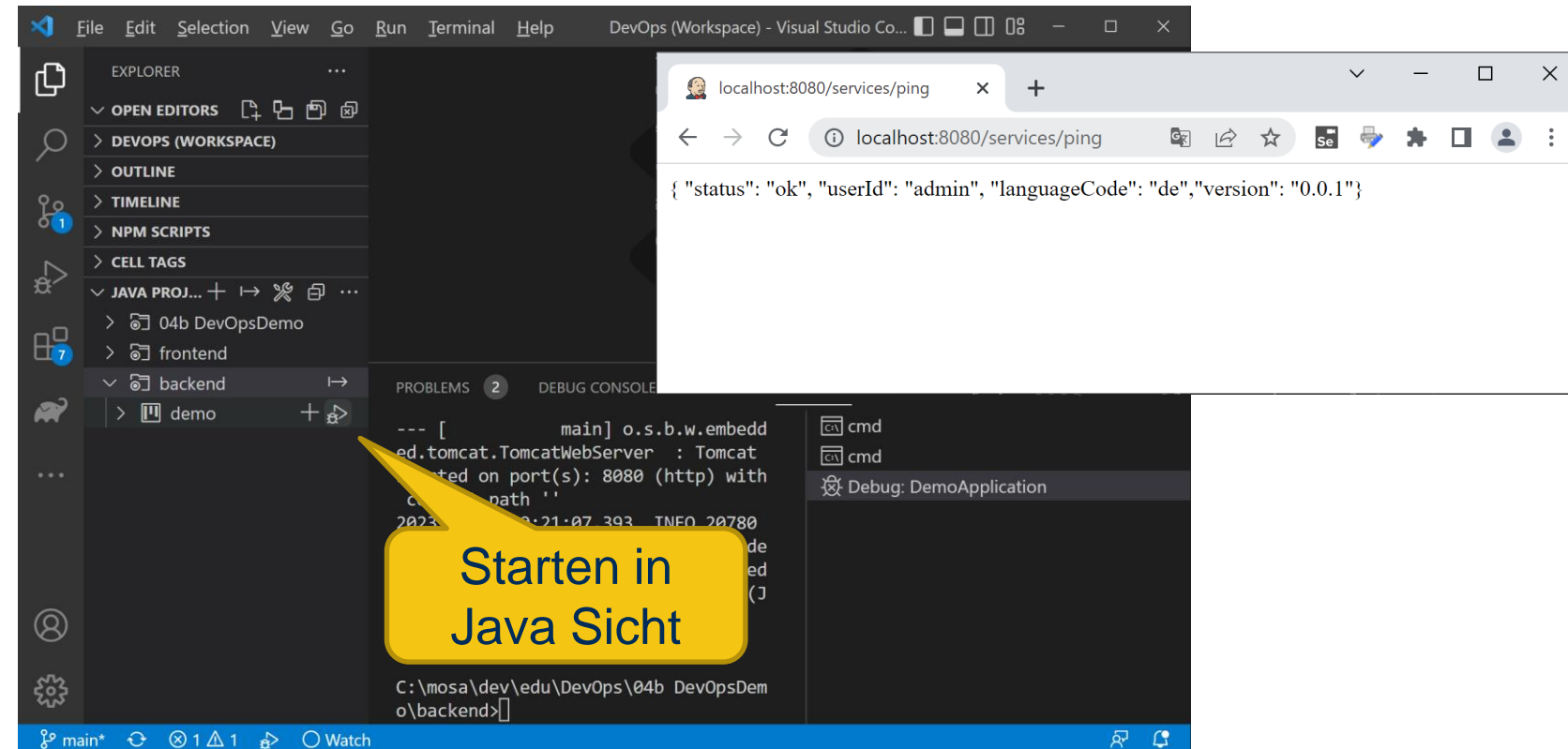
Test: Im Browser <http://localhost:8080/services/ping>

Achtung: Vor dem Starten muss ein bereits laufendes Backend **beendet** werden!

Backend starten und prüfen (mit VS Code)

Tutorial

Das Backend kann mit **Visual Studio Code** gestartet werden:



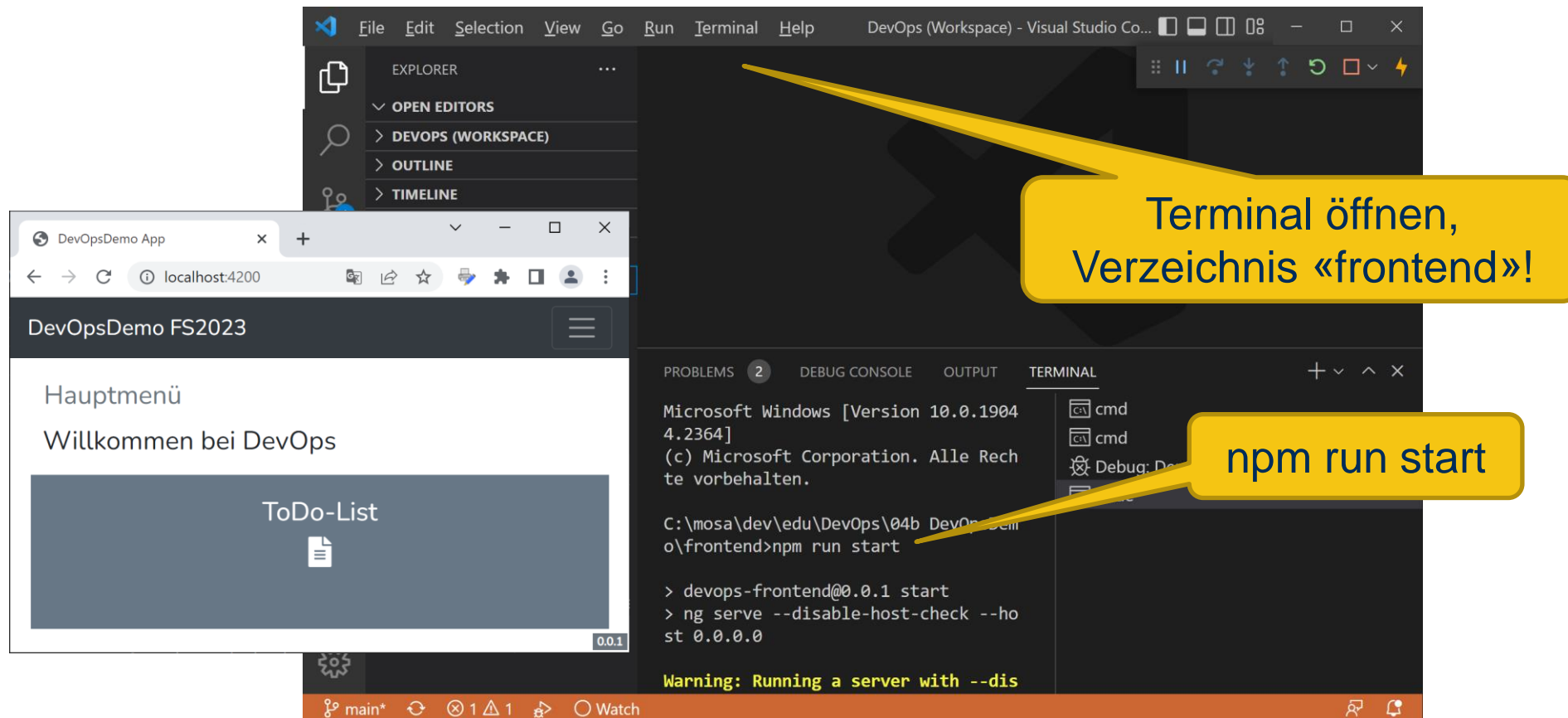
Test: Im Browser <http://localhost:8080/services/ping>

Achtung: Vor dem Starten muss ein bereits laufendes Backend **beendet** werden!

Frontend starten und prüfen

Tutorial

Im Verzeichnis **frontend** mit dem Terminal-Befehl **npm run start**



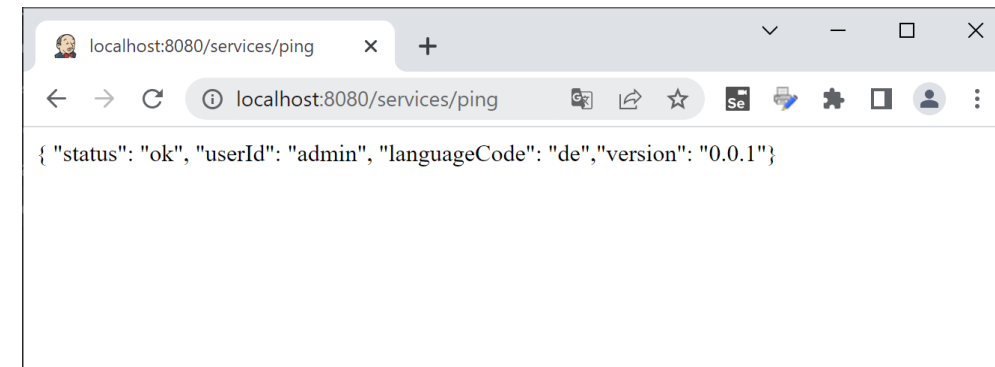
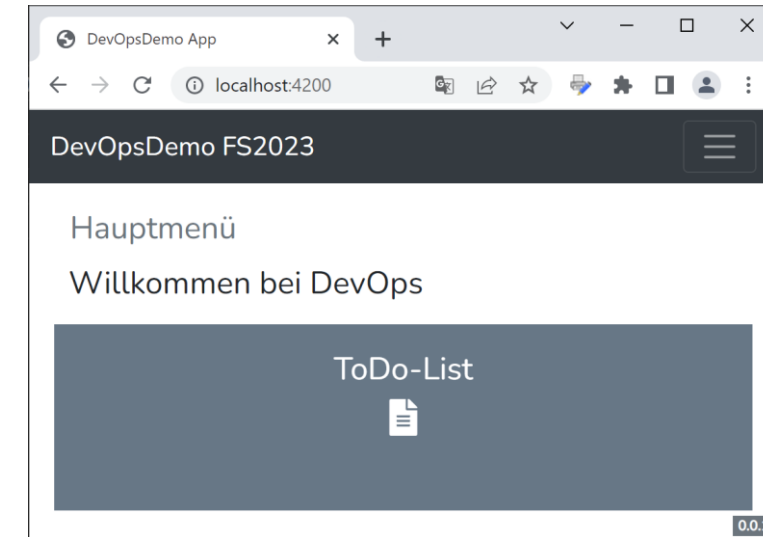
Test im Browser (Build abwarten): <http://localhost:4200/>

Backend

- 8080: Backend

Frontend

- 4200: Web-Applikation

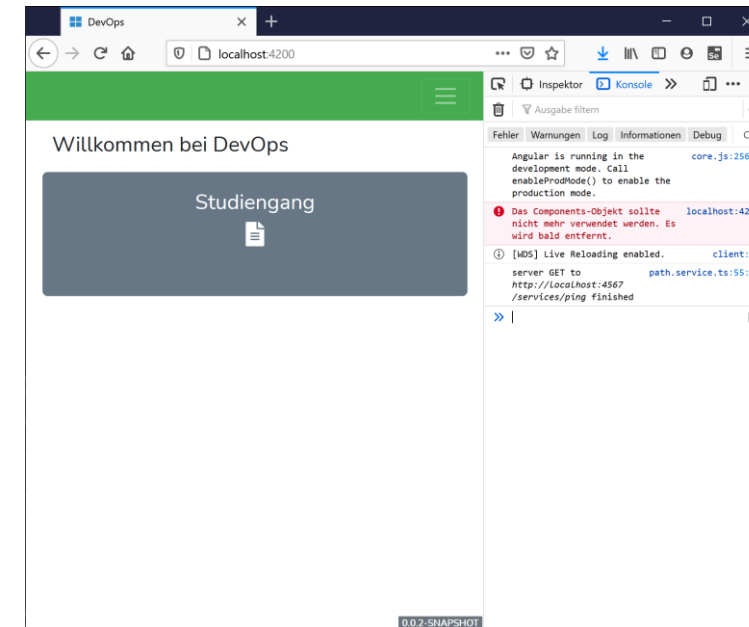


Port bereits besetzt

- Prozesse in Visual Studio Code beenden (Papierkorb-Icon)
- Visual Studio Code beenden und alle java.exe/node.exe Prozesse beenden

Fehler bei Ausführung des Demo-Programmes

- F12: Developer-Sicht im Browser



DevOpsDemo Frontend anpassen: Tutorial

Einfache Anpassungen an DevOpsDemo machen

Kachel hinzufügen

GUI-Model anpassen

- Kachel hinzufügen
- Farben: <https://talkslab.github.io/metro-bootstrap/components.html#tiles>
- Icons: <https://fontawesome.com/icons?d=gallery&m=free>

```
{  
    "type": "button",  
    "name": { default: "Module" },  
    "icon": "fa-file-alt",  
    "color": "wet-asphalt",  
    "page": "modulePage",  
    "width": 2,  
},
```

Page hinzufügen

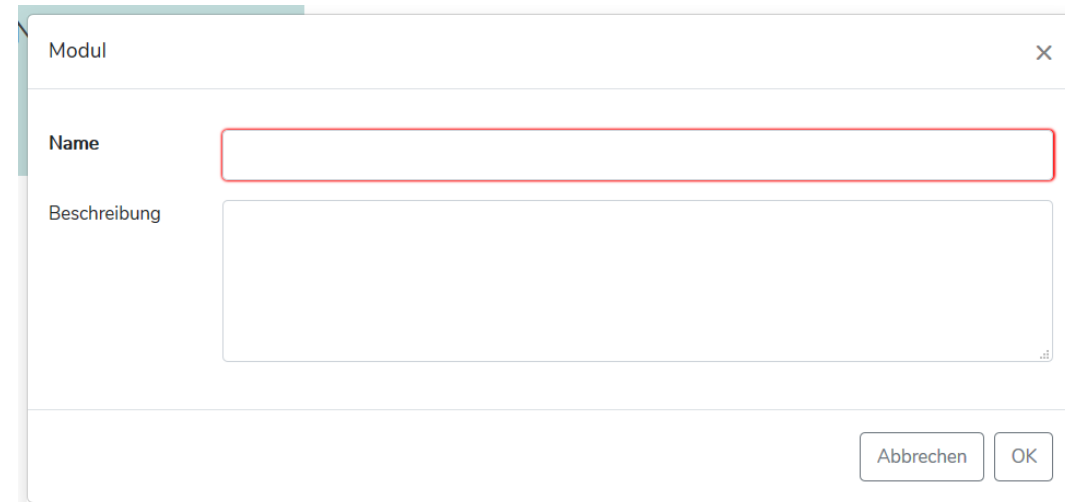
Page mit einer Kachel

```
{
  "id": "modulePage",
  "elementList": [
    {
      "type": "backbutton",
    },
    {
      "type": "newButton",
      "name": { default: "Neues Modul" },
      "icon": "fa-user",
      "color": "green",
      "width": 2,
      "form" : {
        "form" : "ModuleForm"
      }
    }
  ]
}
```

Form hinzufügen

Form mit passender ID
Form-Felder in formFieldList

```
{  
  "id": "ModuleForm",  
  "title": "Module",  
  "formFieldList": [  
    ...  
  ],  
},
```



The image shows a dialog box titled 'Modul' with a close button (X) in the top right corner. Inside the dialog, there are two input fields: 'Name' and 'Beschreibung'. The 'Name' field is a single-line text input, and the 'Beschreibung' field is a multi-line text area. At the bottom right of the dialog, there are two buttons: 'Abbrechen' (Cancel) and 'OK'.

Lernjournal

Ziele

- DevOpsDemo ist geklont
- Der Klon läuft lokal auf dem Rechner des Studierenden
- Git und Build-Technologien können auf DevOpsDemo angewendet werden

Checkliste

- ✓ Klon von DevOpsDemo erstellt
- ✓ Dokumentation lokaler, erfolgreicher Installation auf eigenem Rechner
- ✓ Erweiterungen in Frontend vornehmen (z.B. neue Kachel)

Ziele

- Mit Spring Boot REST-Services erstellen können

Checkliste

- ✓ Als Grundlage wird der Klon von DevOpsDemo verwendet
- ✓ Verschiedene REST-Services (GET, POST, PUT, DELETE) implementieren
- ✓ REST-Services (GET) sind im Browser verfügbar
- ✓ REST-Services sind in Postman verfügbar
- ✓ Resultat auf GitHub pushen