Web App vs. Mobile App - gesammelte Berufserfahrungen Update III

LVA Anwendungen in Wirtschaft und Technik SS 2019

Inhalt

- ➔ Einleitung
- ➔ Vorstellung der Personen
- ➔ Web App
- ➔ Three Tier Architectures
- ➔ Mobile App
  - ◆ Challenges
  - ◆ Angular, RxJS
  - ◆ REST API, JSON
- ➔ Gesammelte Berufserfahrungen

Das Team

- Mag. Ulrike Walch
- 1998 Studium der Wirtschaftsinformatik in Wien
- 20 Jahre Berufserfahrung
- https://www.xing.com/profile/Ulrike_Walch

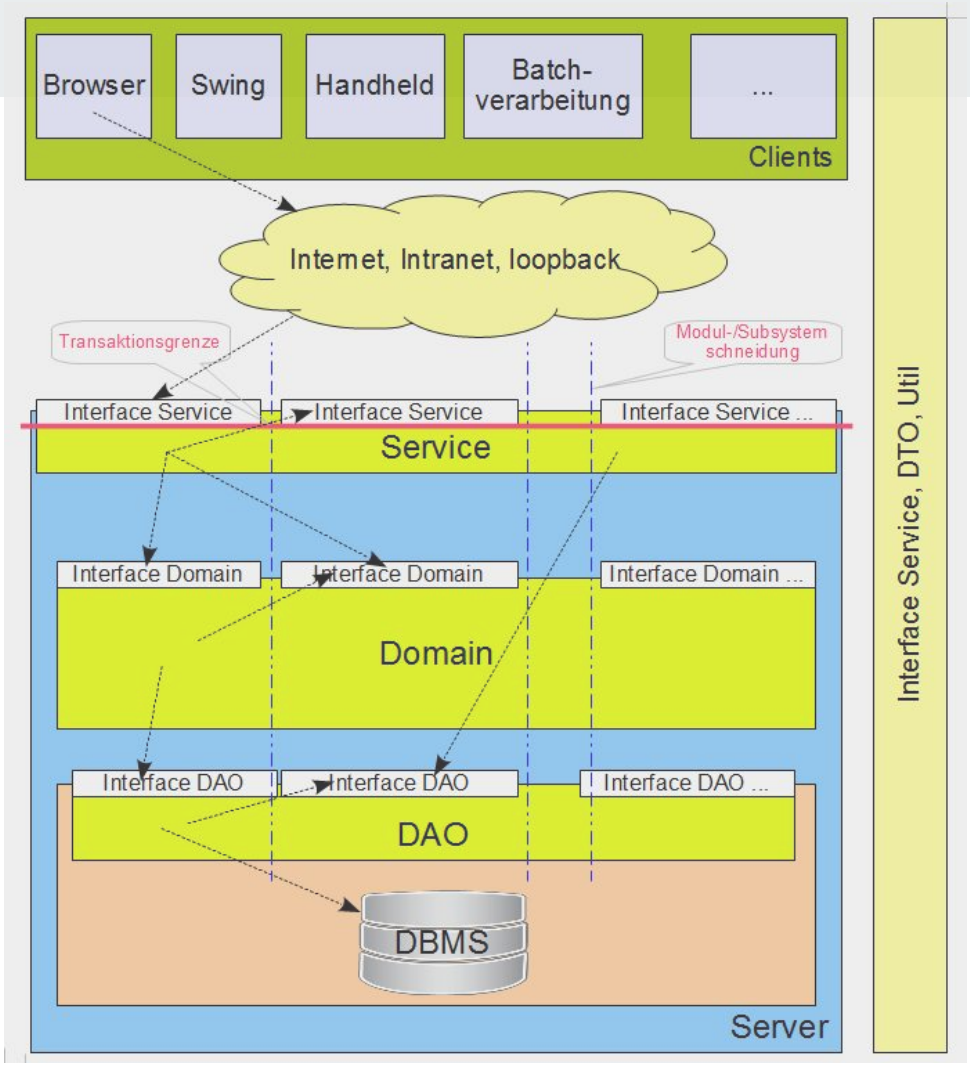
- DI Josef Ornetsmüller
- 1989 Studium der Informatik in Linz
- 30 Jahre Berufserfahrung
- https://www.xing.com/profile/Josef_Ornetsmueller

Webapp

3 - Tier

Three Tier Architecture

Lossless coupling!



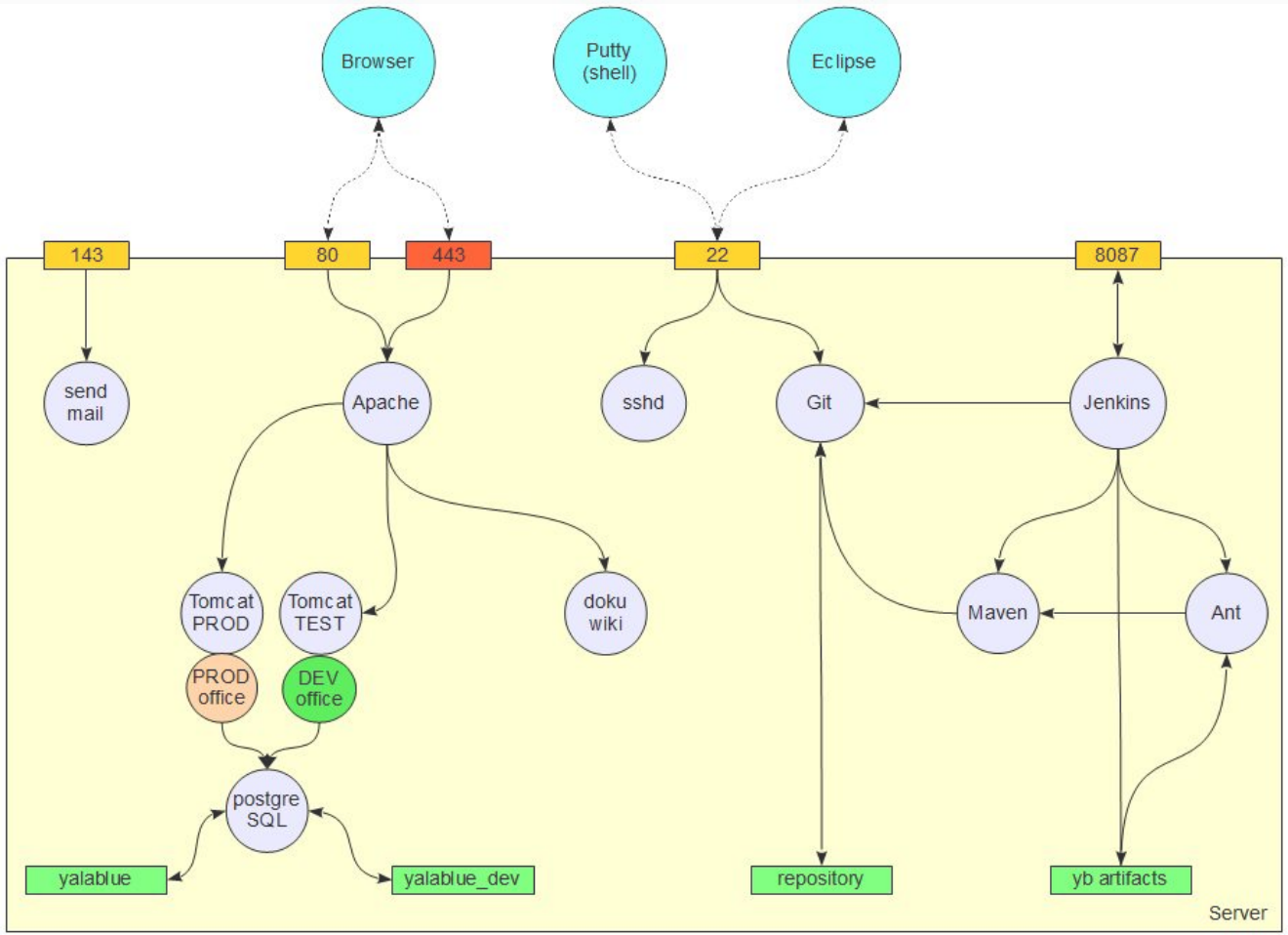
Development/Build 1/2

- ➔ Lokal/interaktive: Laptop, Eclipse, ...
- ➔ Remote/batch: Jenkins, ...
- ➔ Build und Test laufen schichtweise ab DAO, Domain, Service; Client
- ➔ Jenkins, Bamboo, Cruisecontrol, Ant, Maven
- ➔ Artefakte wie EAR-, WAR- JAR-files, Reports compiled, gefüllte Datenbanken, Javadoc, ... werden erzeugt
- ➔ Toolbox Entwickler: npm, Ant, Maven, Javadoc
- ➔ Git, SVN, CVS,
- ➔ Code coverage
- ➔ Staging: DEV, TEST, QA, PROD
- ➔ DokuWiki
- ➔ LIQUIBASE
- ➔ Code coverage

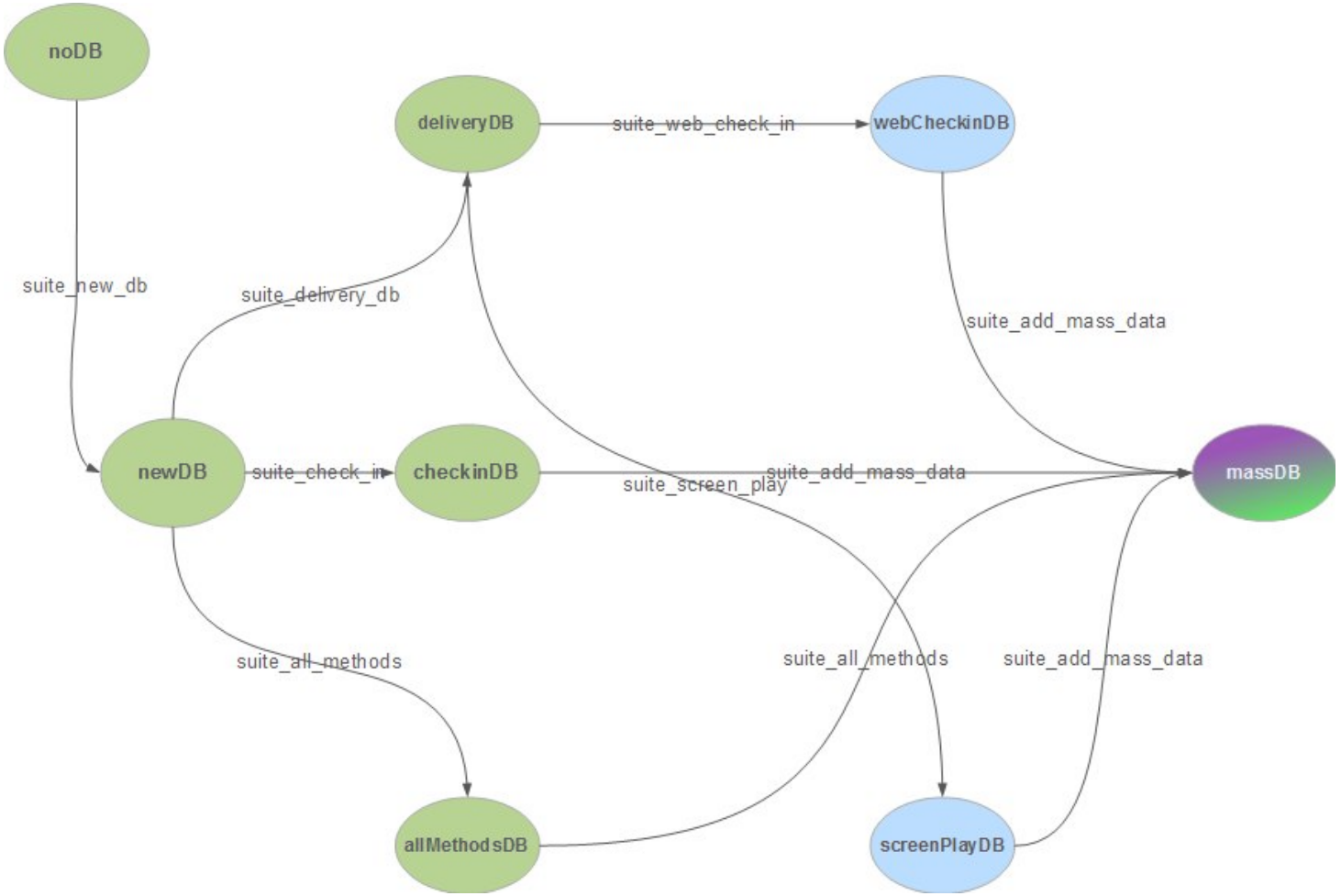
Development/Build 2/2

- ➔ Eclipse; PlugIn: JavaEE Tools table<->class
- ➔ Visual Studio Code: JS
- ➔ Apache
- ➔ Tomcat, TomEE
- ➔ JBoss, Wildfly, Thorntail
- ➔ postgreSQL, ORACLE, DB2, Informix, H2...
- ➔ NoSQL (e.g. Cassandra on Apache Mesos)
- ➔ Putty, Telnet, ...
- ➔ VPN
- ➔ Skype: Teambesprechungen
- ➔ Robocopy

Buildserver Akteure



Test


noDB
deliveryDB
suite_web_check_in
webCheckinDB
suite_new_db
suite_delivery_db
suite_add_mass_data
newDB
suite_check_in
checkinDB
suite_add_mass_data
suite_screen_play
massDB
suite_all_methods
suite_all_methods
suite_add_mass_data
allMethodsDB
screenPlayDB

3rd Party Software

- JDK; AdoptOpenJDK, Amazon Corretto, Azul Zulu
- POI
- Spring
- Wicket
- guava
- Jasper Reports
- Hibernate/JPA, Lucene
- Data Nucleus/JDO, JPA
- JUnit, TestNG
- Log4J(2), logback
- Mailing
- JQuery: Calendar, Ed...
- Webix
- Connection Pooling

Bugfix/Analyse

- ➔ Google Analytics
- ➔ HTML, CSS, JavaScript Debugger
- ➔ Google Chrome, F12, Inspektor; Firefox, Firebug; Internet Explorer, DOM Explorer
- ➔ Eclipse Debugger, remote debugging
- ➔ Logfile analyse
- ➔ jvisualvm, heap, threads, ...
- ➔ Wicket/?: debug mode
- ➔ Eigene eingebaute Tools (Cockpit)
- ➔ Methoden Laufzeitanalyse
- ➔ Logger auf DEBUG,. sonst INFO
- ➔ Glowroot

Mobile App

Angular - RxJS - REST

Mobile App: Challenges

- ➔ Kennzeichen klassischer Browser-Anwendungen für große Business Anwendungen:
  - ◆ großes Datenvolumen und komplexe Workflows
  - ◆ aufwendiges User Interface
- ➔ Wenn man eine Web App für Mobile Devices adaptieren möchten, steht man vor speziellen Herausforderungen:
  - ◆ Viele verschiedene Hersteller, Betriebssysteme, Betriebssystem Versionen
  - ◆ Geringe Bandbreiten, Netzverfügbarkeit, geringe Devicegrößen
  - ◆ Anpassung des Designs, wieviele Daten können sinnvoll dargestellt werden?
  - ◆ Anpassung des Workflows aufgrund der geringen Devicegrößen

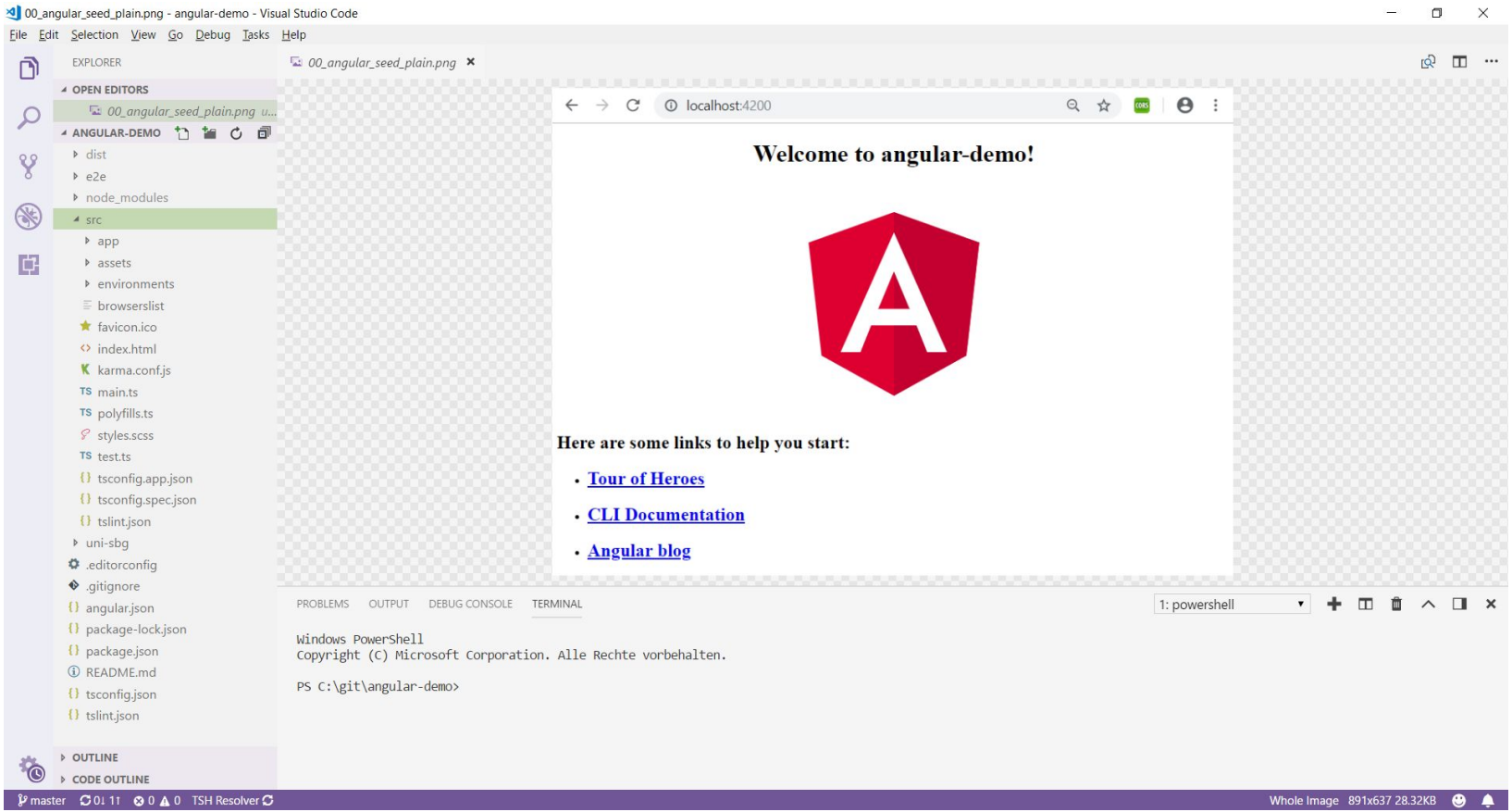
Mobile App: Grundsätzliche Entscheidungen

- ➔ Browser-Anwendung portieren ja/nein?
- ➔ Wenn Mobile App ja:
  - ◆ Zielgruppe, Zielplattform?
  - ◆ Native oder hybride App oder PWA?
  - ◆ PWA: Progressive Web App
    https://www.heise.de/developer/artikel/Faktencheck-zu-Progressive-Web-Apps-Teil-1-Plattformen-Plug-ins-4259135.html
- ➔ Native: Technologie-Stack durch die Plattform vorgegeben
- ➔ Hybrid: HTML5/CSS3 mit native features via Cordova/PhoneGap
- ➔ PWA: Bootstrapping einer Web App via Service Worker als Mobile App

Technology example: Angular

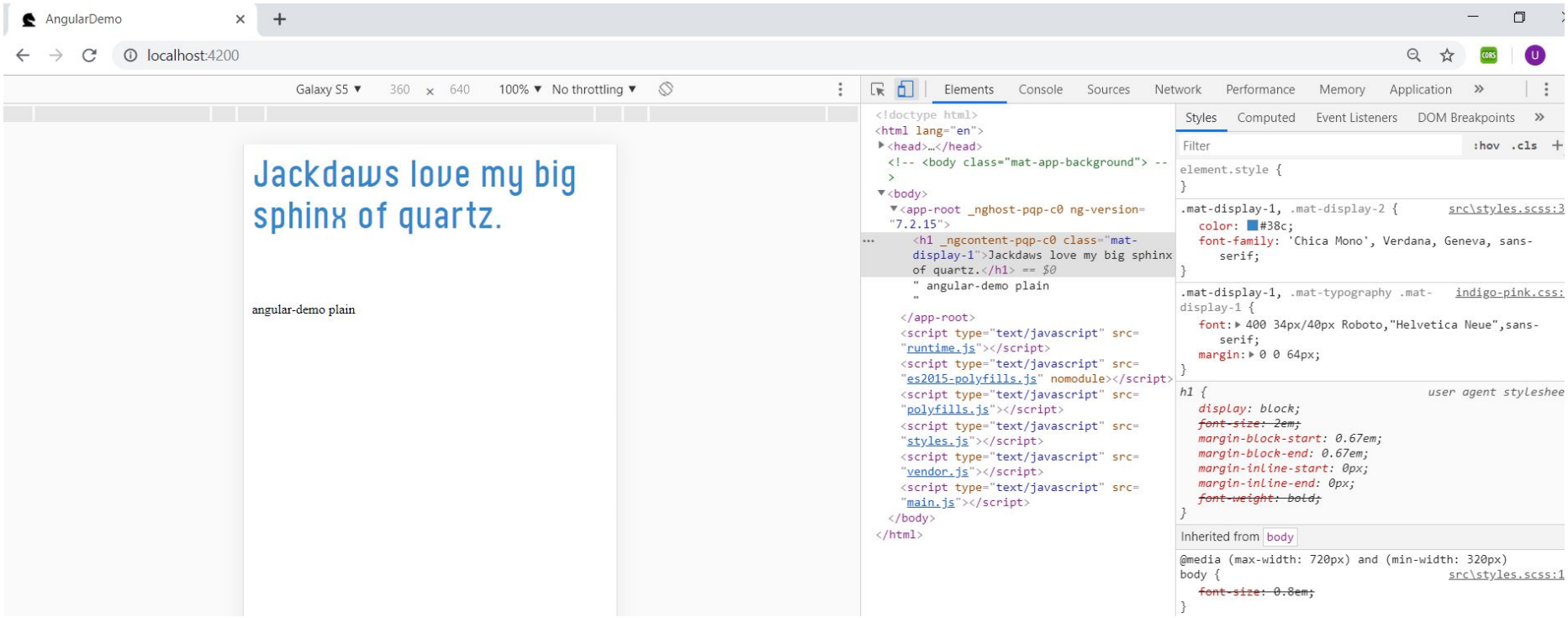
Angular - RxJS - TypeScript
REST API - JSON - C3 Charting

Example project setup: Visual Studio Code



Einrichten eines initialen Projekts

- ➔ Install Node.js, NPM (je nach Plattform).
- ➔ Install Angular CLI, create new project (TypeScript)
  - ◆ Config: package.json, angular.json.
  - ◆ npm packages for API doc generation (compodoc), i18n (ngx-translate)
  - ◆ VSC extensions (Prettier, Move TS, ...)
  - ◆ Config: karma.conf.js; prepare for Jenkins integration
  - ◆ Add Angular Material (customize theming and typography)
  - ◆ Add Angular routing for navgation & modularization


AngularDemo
localhost:4200
Galaxy S5 360 x 640 100% No throttling
Jackdaws love my big sphinx of quartz.
angular-demo plain
Elements Console Sources Network Performance Memory Application
<!doctype html>
<html lang="en">
<head>…</head>
<!-- <body class="mat-app-background"> -->
<body>
<app-root _nghost-pqp-c0 ng-version="7.2.15">
<h1 _ngcontent-pqp-c0 class="mat-display-1">Jackdaws love my big sphinx of quartz.</h1> == $0
" angular-demo plain "
</app-root>
<script type="text/javascript" src="runtime.js"></script>
<script type="text/javascript" src="es2015-polyfills.js" nomodule></script>
<script type="text/javascript" src="polyfills.js"></script>
<script type="text/javascript" src="styles.js"></script>
<script type="text/javascript" src="vendor.js"></script>
<script type="text/javascript" src="main.js"></script>
</body>
</html>
Styles Computed Event Listeners DOM Breakpoints
Filter :hov .cls +
element.style { }
.mat-display-1, .mat-display-2 { src\styles.scss:3
color: #38c;
font-family: 'Chica Mono', Verdana, Geneva, sans-serif;
}
.mat-display-1, .mat-typography .mat-display-1 { indigo-pink.css:
font: 400 34px/40px Roboto,"Helvetica Neue",sans-serif;
margin: 0 0 64px;
}
h1 { user agent styleshee
display: block;
font-size: 2em;
margin-block-start: 0.67em;
margin-block-end: 0.67em;
margin-inline-start: 0px;
margin-inline-end: 0px;
font-weight: bold;
}
Inherited from body
@media (max-width: 720px) and (min-width: 320px)
body { src\styles.scss:1
font-size: 0.8em;
}

Round-Trip vs. Single-Page Application

- ➔ Round trip:
  - ◆ Klassische Web-Anwendung, die auf dem HTTP-Protokoll basiert (Request / Response)
- ➔ Single page:
  - ◆ Initiales HTML-Dokument, Views werden dann über AJAX häppchenweise geladen

Example: Angular single page

- ➔ Single Page: index.html
- ➔ View change via Angular routing mechanism

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>AngularDemo</title>
    <base href="/">

    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500" rel="stylesh
  </head>

  <body>
    <app-root></app-root>
  </body>
</html>
```

Angular & RxJS

- RxJS - ReactiveX for JavaScript
  - “An API for asynchronous programming with observable streams.”
- JavaScript = single threaded
- einzelne Teile werden im View via Ajax “eingeblendet”
- Asynchronous example: Anbindung einer REST API
  - REST API service liefert ein Observable:

    “Hier hast du das Versprechen, dass ein Ergebnis zurückkommt, aber ich kann Dir nicht sagen wann und ob es ein gültiges Ergebnis oder ein Fehler sein wird.”

Beispiel REST API

- ➔ REST = Representational State Transfer
- ➔ Architekturstil für Web Services
- ➔ Fokus auf Sicherheit & Skalierbarkeit

RxJS example: Generate c3 chart

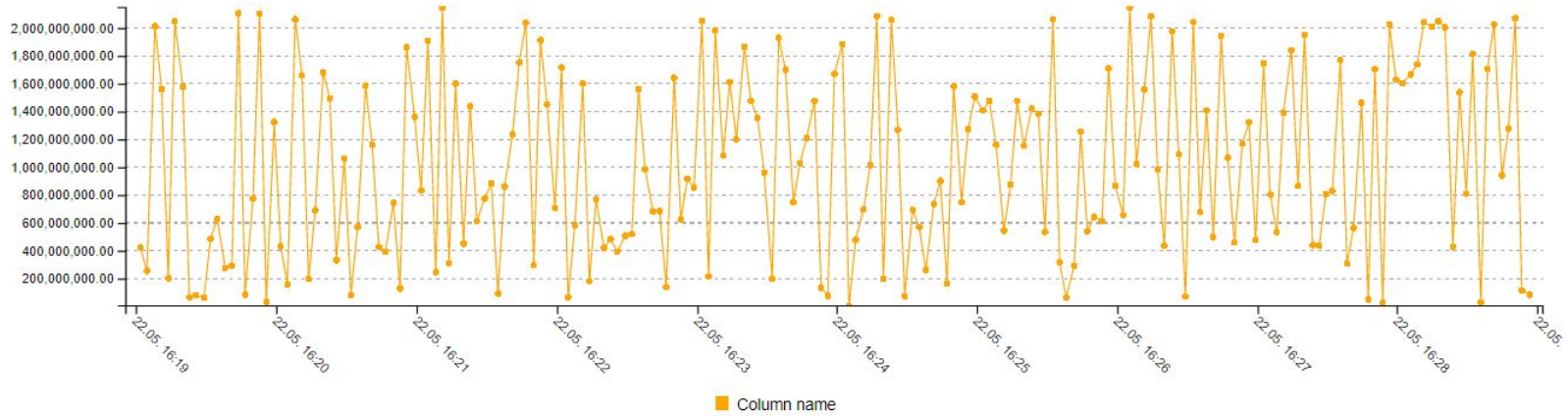
```
private loadChart() {
  this.restApiService.getTimeseriesDatapoints('device-1', new Date().getTime(), new Date().getTime())
    .pipe(
      // use async pipe in view or explicitley unsubscribe
      untilComponentDestroyed(this)
    )
    .subscribe((tdps: TimeseriesDatapoints) => {
      console.log('loadChart', 'number of datapoints=', tdps[0].datapoints.length);

      const chartConfig = this.gsC3Service.getChartConfig(this.chartId, tdps[0].datapoints);

      this.chart = c3.generate(chartConfig);
    },
      error => {
        // e.g. http error, network error, ...
        console.error(
          'loadChart failed'
        );
      });
}
```

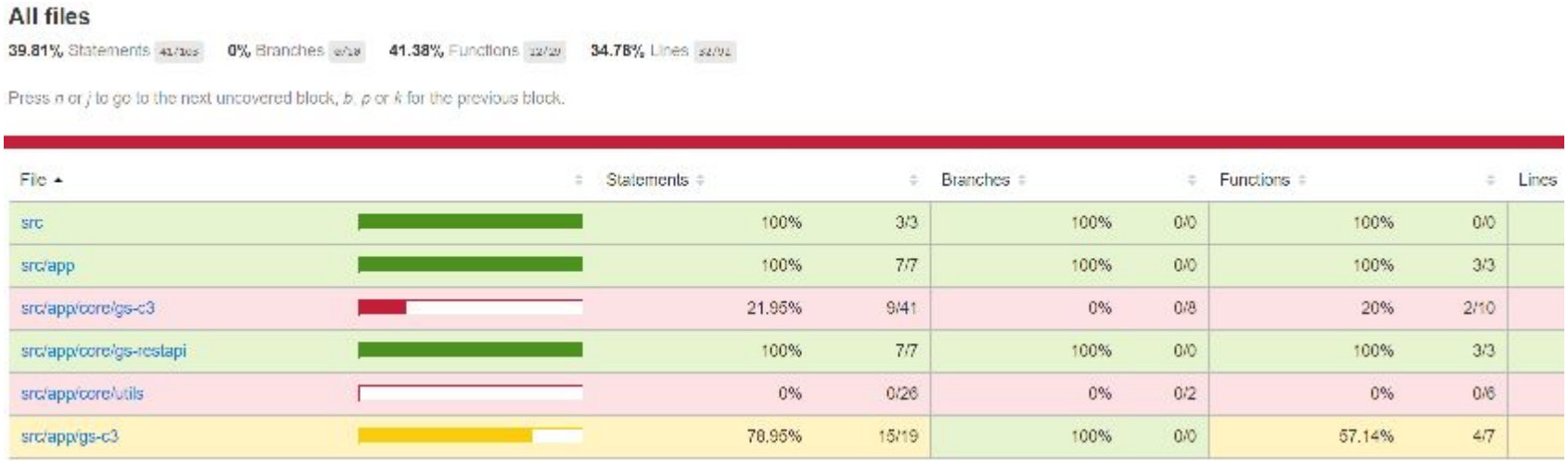
Angular app example: Generate c3 chart

angular | c3



TDD: Test-driven development

1. Change implementation detail, developer test & debug
2. Execute unit test & fix.
3. Execute e2e test & fix.



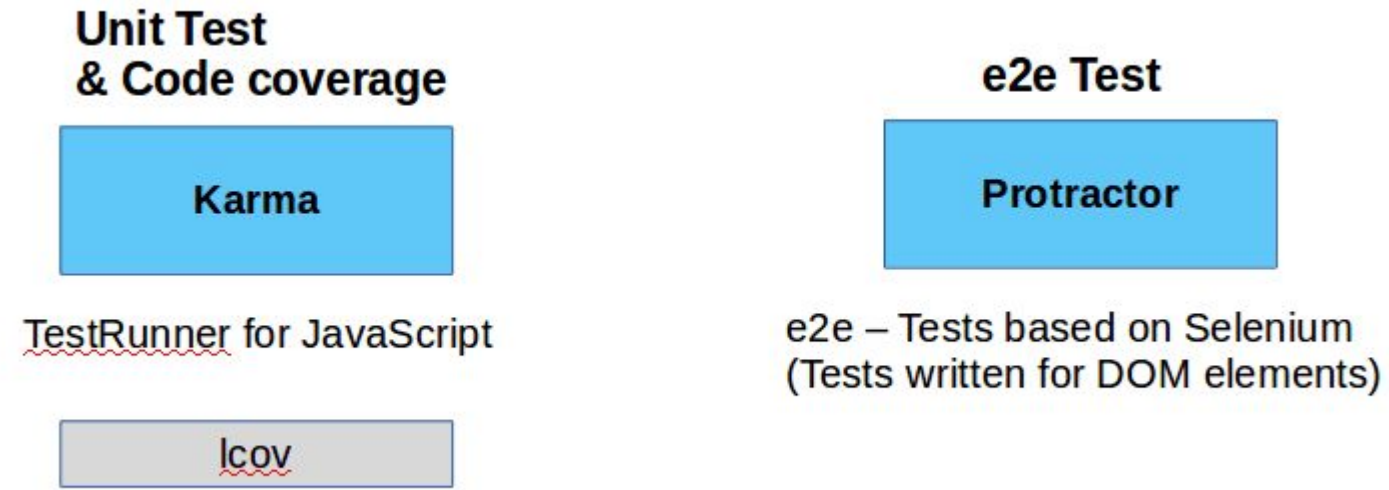
All files

39.81% Statements **0%** Branches **41.38%** Functions **34.78%** Lines

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines
src	100%	3/3	100%	0/0	100%	0/0	
src/app	100%	7/7	100%	0/0	100%	3/3	
src/app/core/gs-c3	21.95%	9/41	0%	0/8	20%	2/10	
src/app/core/gs-restapi	100%	7/7	100%	0/0	100%	3/3	
src/app/core/utils	0%	0/26	0%	0/2	0%	0/6	
src/app/gs-c3	78.95%	15/19	100%	0/0	57.14%	4/7	

TDD: Testing an Angular application



Gesammelte Berufserfahrungen

Update

Zuerst kommt... (J)

...das Wichtige dann das Richtige

...das Richtige dann das Wichtige

Ihre Zukunft (J)

- ➔ 7 €/Monat für Ihre Zukunft
- ➔ Sie sind anderer Meinung
- ➔ Grenzen Sie sich ab vs. ‘Staubsauger’
- ➔ Ihre Idee findet keine Anerkennung, sie sind…
- ➔ Wie Viele neue ‘Brocken’ packt das Projekt
- ➔ Firmentyp/Manager: Ich darf alles vs. kaum was
- ➔ 6 Monate Schonfrist, hat die neue Firma
- ➔ Wie motivieren sie sich für den nächsten Tag?
- ➔ SCRUM, KANBAN
- ➔ Intrapreneurship/Binnen-unternehmertum

Technische Skills 1 (J)

- ➔ Software Design: Pattern, Tiers
- ➔ Domain Driven Design; Test Driven Design
- ➔ Programmiersprachen
- ➔ Betriebssysteme: Windows, Unix/Linux
- ➔ Ihre UNIX-Box im Internet verwalten sie selbst
- ➔ Staging: DEV, TEST, QA, PROD, ...
- ➔ Datenbanken (SQL, DBMS)
- ➔ Regular Expression

Technische Skills 2 (U)

- CVS, SVN, Git
- Hype: SCRUM
- Notepad++
- Server-Technologien:
  - J2EE EJB
  - Spring, Hibernate, JPA, JDO
  - Node JS
- Client-Technologien:
  - JSP, JSF, Wicket
  - HTML, CSS, JavaScript
  - Angular/React/Ember/Vue/...
  - Cordova/Phone Gap
  - Progressive Web App

Soft Skills 1 (J)

- ➔ KISS
- ➔ Focus on point
- ➔ Fragen stellen
- ➔ Teamfähigkeit
- ➔ Veranschaulichung von Problemen und Lösungsansätzen
- ➔ Humor
- ➔ Kaizen
- ➔ Artifact über CRUD nachdenken

Soft Skills 2 (U)

- ➔ Kreativität
- ➔ Durchstich
- ➔ Kundenkontakt
- ➔ Einarbeitung in Fachlichkeit (Domain)
- ➔ Probleme in Arbeitspakete zerlegen
- ➔ Am falschen Weg umkehren
- ➔ Das gemeinsame Ziel verfolgen

Dokumentation (J)

- ➔ JavaDoc sinnvoll nutzen
- ➔ Nahe am Code
- ➔ Tools generieren automatisch
- ➔ Sie lesen viel Code
- ➔ Sie finden einen alten Dokumentationen Friedhof
- ➔ Unwartbar; Zuviel Aufwand
- ➔ Confluence und JIRA
- ➔ Übertreiben Sie nicht
- ➔ Token – Querbeet
- ➔ Literate Programming
- ➔ Die Idee
- ➔ Was dem nächsten Leser hilft

Up-to-Date bleiben 1 (J)

- ➔ Konzeptwissen Halbwertszeit von 10 bis 15 Jahren, Produktwissen anderthalb bis zwei Jahren
- ➔ Lesen lesen lesen; Video schauen
- ➔ Konferenzen besuchen
- ➔ Flexibilität bewahren
- ➔ Zertifizieren Sie sich

Up-to-Date bleiben 2 (J)

- ➔ Nützen Sie die Gelegenheit zu Technologieevaluierungen, Tutorials
- ➔ Fragen Sie Kollegen, profitieren Sie von den Fehlern, die andere schon gemacht haben
- ➔ Wie nahe bin ich am Puls der aktuellen Informatik Technologie/Technik?
- ➔ Bin ich an einem Toten Ast?

- ➔ Verliere ich mit jedem Tag Informatik-‘Wissen’?

  Was passiert dann bei einem Firmenwechsel?

Programmier-Alltag 1 (U)

- ➔ Wie schnell kann ich den Fehler reproduzieren?
- ➔ Wie schnell bin ich an der „frisch“ zu programmierenden Stelle?
- ➔ Wie lange brauche ich um eine „Seite“ Code zu verstehen?
- ➔ Wer darf nicht auf Urlaub gehen, damit ich weiterarbeiten kann?
- ➔ Bin ich von externen Systemen abhängig (DB, Server, ...) oder kann ich autark arbeiten (lokale IDE)?
- ➔ Ist fachliche Ansprechpartner verfügbar?

Programmier-Alltag 2 (J)

- ➔ Code mit den „sprechenden“ Variablenname, klar strukturiert?
- ➔ Ist der Code noch ‘refactor fähig’?
- ➔ Kann ich sorgenfrei Bugfixing - einen Feature Request abarbeiten?
- ➔ Habe ich überwiegend Spaß an der Arbeit?
- ➔ Gibt es einen letzten Breakpoint?
- ➔ Wie finde ich mich in der Package Struktur zurecht?
- ➔ Typen: ‘Ich-Weiß-Immer-Alles’; Mitschwimmer; Buddy; ...

A day

coffe, e-mail -, news check

scrum pick; scrum: daily

analyse, design, edit, test java, js, html, css, sql, ...

backlog refinement

pair programming

pool for cooling ;-)

opt: night bug finding session

opt: check new technology

opt: retro, sprint planning, - review

A day II

coffee, email, jenkins build

feature, gedankenintensives, probleme debuggen

-- pause

goto zur abstimmung

daily scrum

-- mittagspause

lightweight feature, issues abarbeiten

opt: grooming, planning, review

opt: deployment test, hotfix

Fragen


9 MARKETING
WE DON'T KNOW WHAT THE PRODUCT SHOULD DO OR WHO WOULD USE IT.
© UFS, Inc.
S. Adams
9 MARKETING