Webapp vs. Mobile App - gesammelte Berufserfahrungen Update II

LVA Anwendungen in Wirtschaft und Technik

Inhalt

- ➔ Einleitung
- ➔ Vorstellung der Personen
- ➔ Web App
- ➔ Three- Tier -Architectures
- ➔ Mobile App
  - ◆ Challenges
  - ◆ Anguar JS
  - ◆ REST API
- ➔ Gesammelte Berufserfahrungen

Das Team

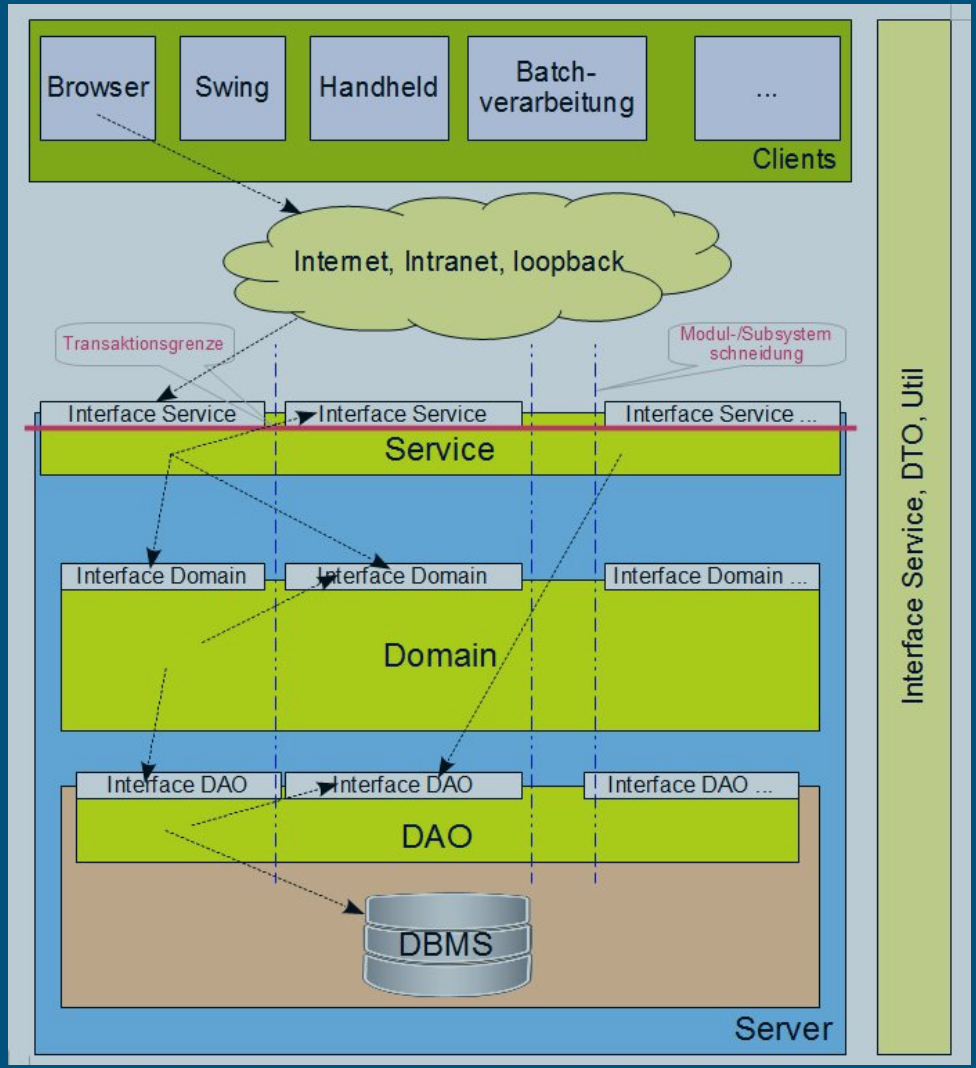
- Mag. Ulrike Walch
- 1998 Studium der Wirtschaftsinformatik in Wien
- 19 Jahre Berufserfahrung

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

Webapp

3-Tier

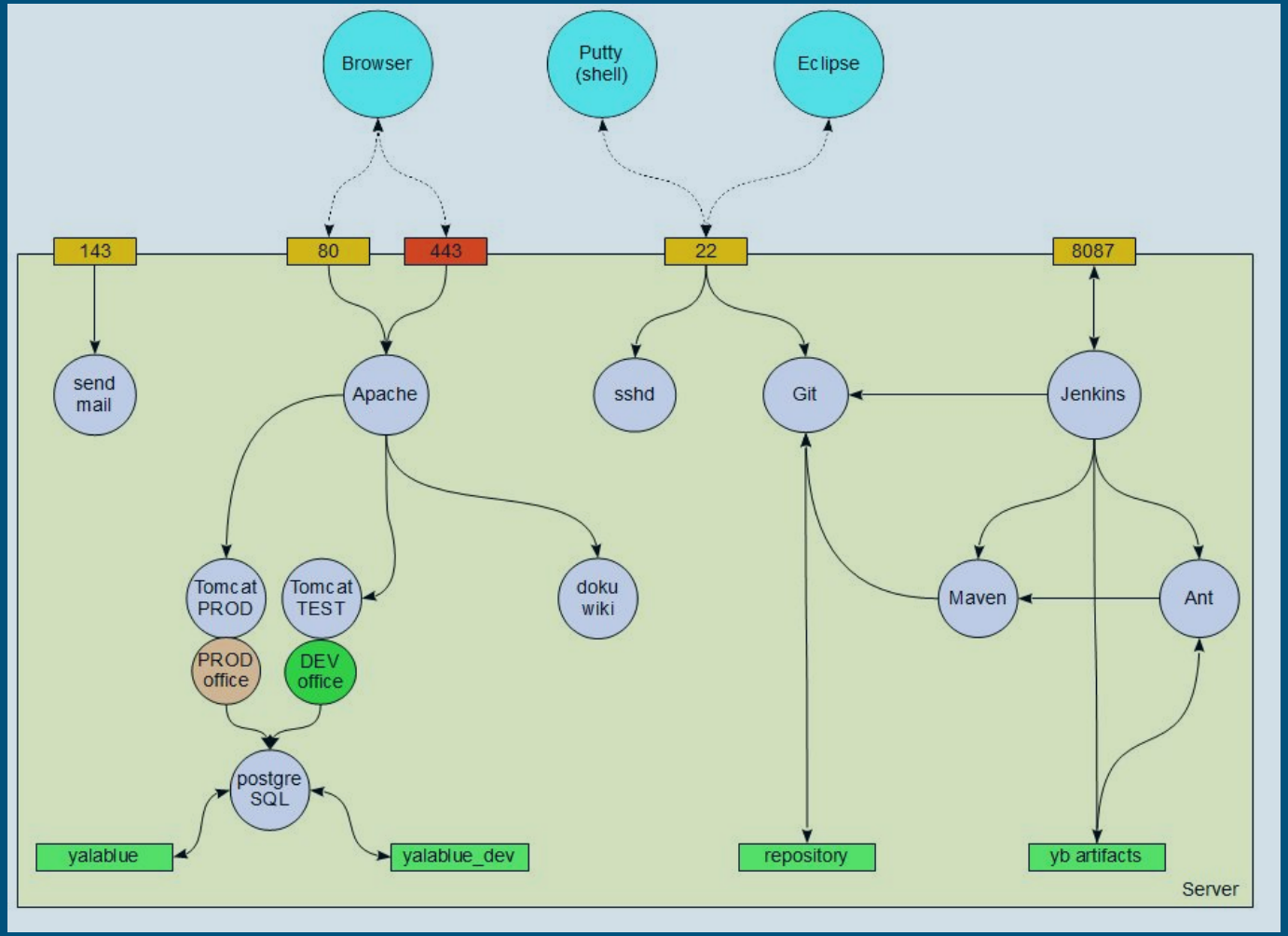
Three Tier Architecture



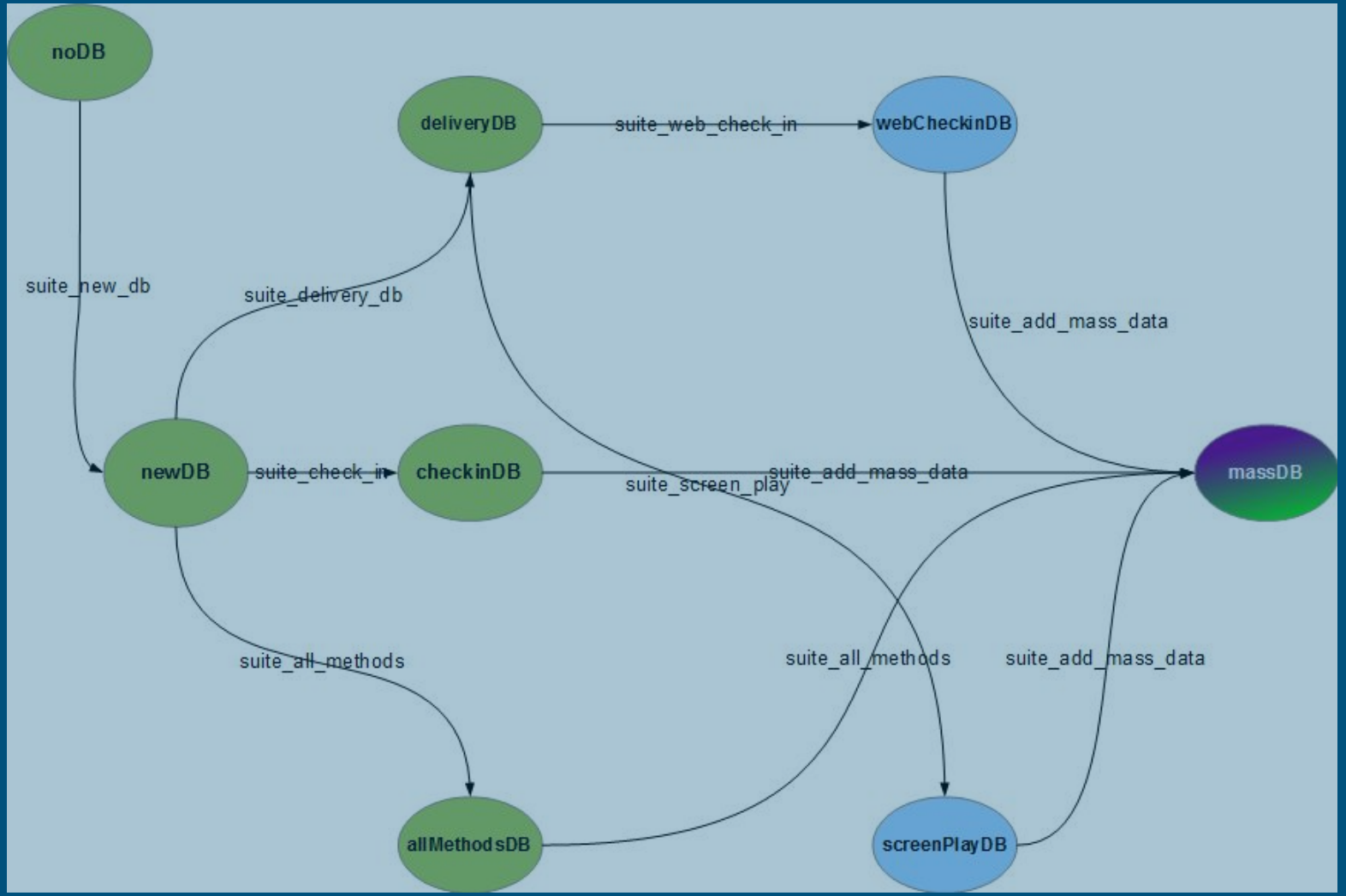
Build

- ➔ 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: Ant/Maven: clean, build, test, Javadoc, native2ASCII, ...
- ➔ Git, SVN, CVS
- ➔ Staging: DEV, TEST, PROD

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 1

- Java SDK
- Apache
- Ant
- Maven
- Connection Pooling
- Code coverage
- Jenkins
- DokuWiki
- Eclipse J2EE
- Google Analytics
- Hibernate/JPA, Lucene
- Data Nucleus/JDO, JPA
- JQuery: Calendar, Ed...
- LIQUIBASE

3rd Party Software 2

- Log4J(2)
- Mailing
- Putty
- Spring
- TestNG
- Tomcat
- Wildfly
- Wicket
- guava
- postgreSQL
- Eclipse
- Jasper Reports

Bugfix/Analyse 1

- ➔ HTML, CSS, JavaScript Debugger; Bildschirmgrößen
- ➔ Google Chrome, Inspektor; Firefox, Firebug; Internet Explorer, DOM Explorer
- ➔ Code coverage

Bugfix/Analyse 2

- ➔ Eclipse Debugger, remote debugging
- ➔ Logfile analyse
- ➔ jvisualvm, heap, threads, ...
- ➔ Wicket: debug mode
- ➔ Eigene eingebaute Tools (Cockpit)

Mobile App

Angular JS, REST API

Mobile App: Challenges

- ➔ Klassische Browser-Anwendungen - große Business Anwendungen
- ➔ Kennzeichen:
  - ◆ großes Datenvolumen, das aufbereitet und dargestellt werden muss
  - ◆ komplexe Workflows und aufwendige User-Interaktionen
  - ◆ aufwendiges User Interface
- ➔ Im einfachsten Fall: Mobile Client für bestehenden Server

Mobile App: Challenges

- ➔ Viele verschiedene Hersteller, Betriebssysteme, Betriebssystem Versionen
- ➔ Geringe Bandbreiten
- ➔ Netzverfügbarkeit
- ➔ Geringe Device Größe:
  - ◆ Wie Viele Daten können sinnvoll dargestellt werden?
  - ◆ Wie kann der Workflow sinnvoll gestaltet werden (User Interaktion)?
  - ◆ Anpassung des Designs
- ➔ Verfügbare Technologien

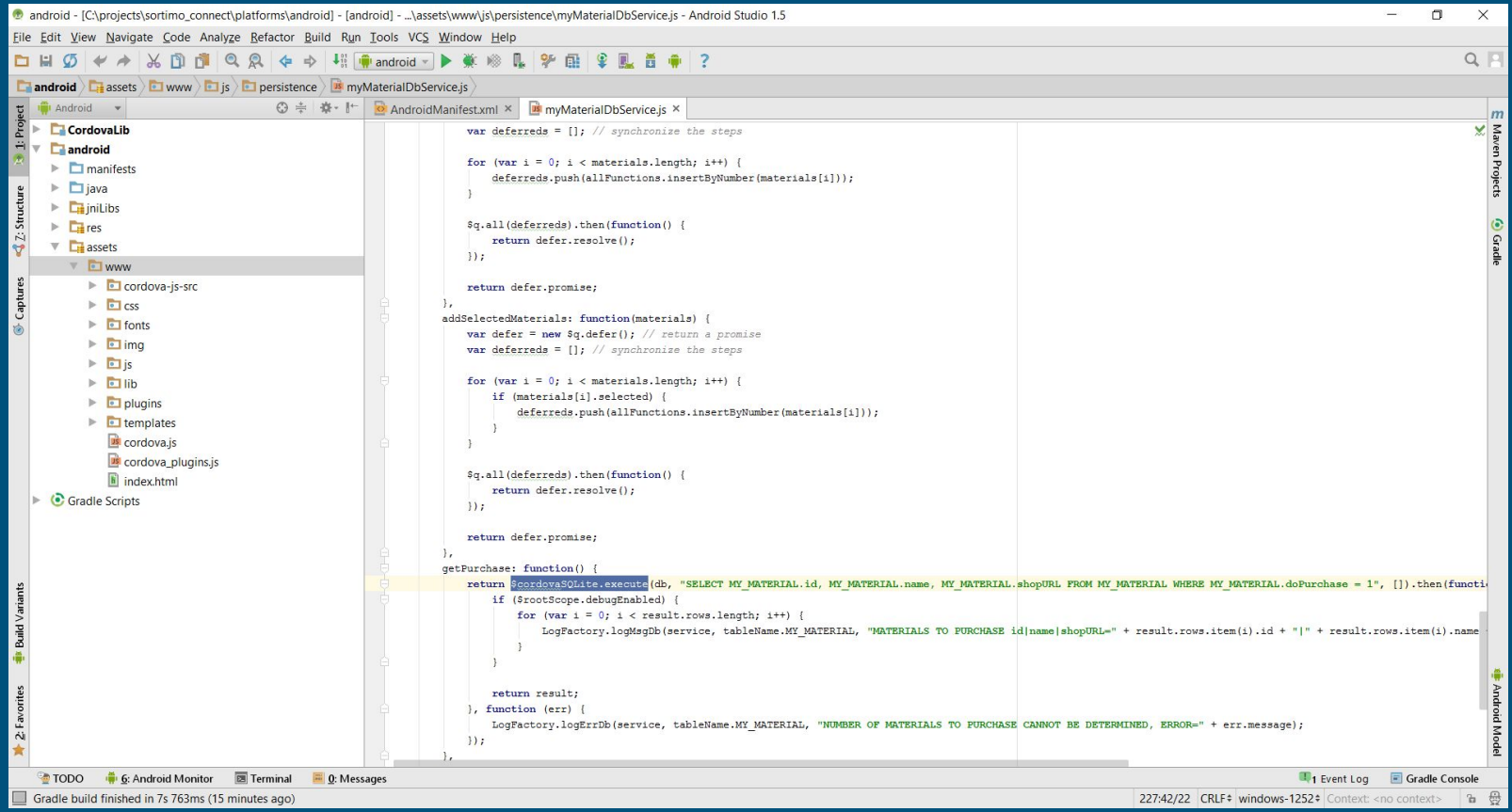
Mobile App: Grundsätzliche Entscheidungen

- ➔ Browser-Anwendung portieren ja/nein?
- ➔ Wenn Mobile App ja:
  - ◆ Zielgruppe, Zielplattform?
  - ◆ Native oder hybride App?
- ➔ Native: Technologie-stack durch die Plattform vorgegeben
- ➔ Hybrid: HTML5/CSS3 mit native features via Cordova/PhoneGap

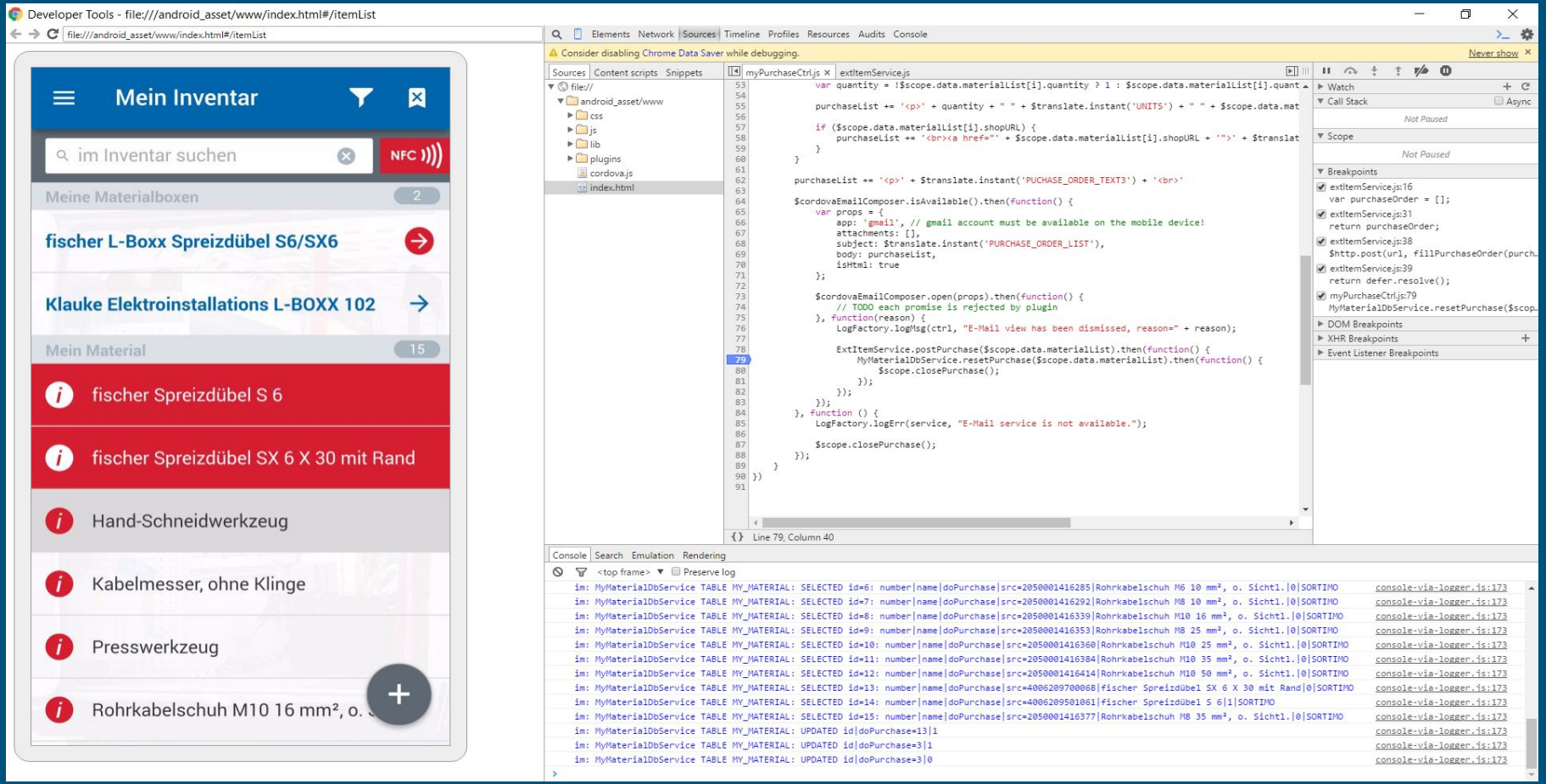
Hybrid Mobile App

Angular JS, REST API

Example: Android Studio



Google Chrome Remote Debugging



Hybrid Mobile App: Technologie-stack

- ➔ Ionic Framework: UI-Framework
- ➔ Angular JS: JavaScript Framework; MVC & Testing
- ➔ Apache Cordova (Adobe PhoneGap): Device abstraction layer
- ➔ SQLite: Relational database on smart devices

= Three-Tier-Architecture

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

Angular JS

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

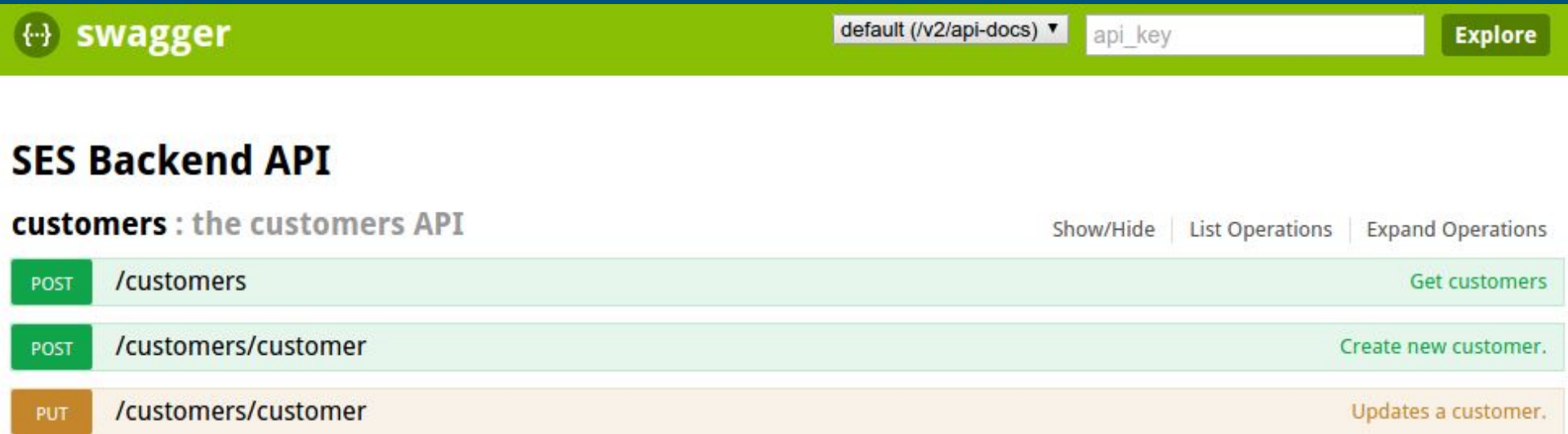
```
        <script src="js/addItem/services/barcodeService.js"></script>
        <script src="js/addItem/services/extItemService.js"></script>
        <script src="js/addItem/services/nfcService.js"></script>

        <script src="js/options/profileCtrl.js"></script>
        <script src="js/options/communityCtrl.js"></script>
        <script src="js/options/appConfigCtrl.js"></script>
    </head>

    <body class="platform-android platform-cordova platform-webview">
        <ion-nav-view></ion-nav-view>
    </body>
</html>
```

Angular JS: Promises

- ➔ API reference $q:
  - ◆ “Service that helps you run functions asynchronously”
- ➔ JavaScript = single threaded
- ➔ einzelne Teile im View können je nach Aufgabenstellung später “eingeblendet” werden
- ➔ Angular Libraries basieren auf dem Prinzip der Promises
- ➔ z.B. REST calls via API reference $http

Beispiel REST API

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



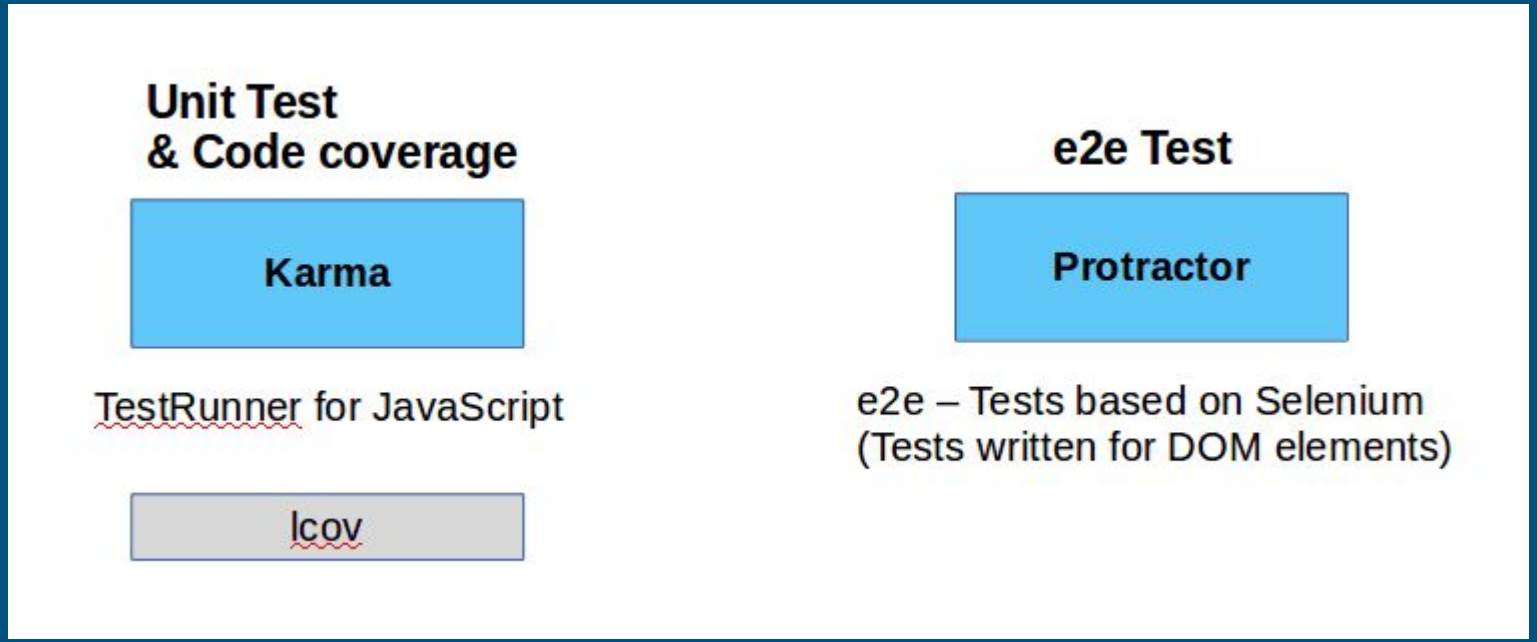
UI: Befüllen einer DropdownBox

- ➔ Angular Service $http liefert ein Promise
- ➔ “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.”

REST endpoint for getting all customers

```
.factory('Customers', function(..) {
    return {
        customers : function() {
            var deferred = $q.defer();

            var url     = /customers;
            var method  = 'POST';

            ServicesHelper.getHttpPromise(url, method).then(function(result) {
                    deferred.resolve(result);
            }, function(reason) {
                    $log.error('Customers.customers: ' + JSON.stringify(reason));

                    deferred.reject([]);
            });

            return deferred.promise;
        }
    };
});
```

Testing a web app based on Angular JS



Gesammelte Berufserfahrungen

Update

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 Hype now?
- ➔ Intrapreneurship/Binnen-unternehmertum

Zuerst kommt... (J)

...das Wichtige dann das Richtige

...das Richtige dann das Wichtige

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, PROD, ...
- ➔ Datenbanken (SQL, DBMS)
- ➔ Regular Expression

Technische Skills 2 (U)

- CVS, SVN, Git
- Hype: SCRUM
- Server-Technologien:
  - J2EE EJB
  - Spring, Hibernate, JPA, JDO
  - Node JS
- Client-Technologien:
  - JSP, JSF, Wicket
  - HTML, CSS, JavaScript
  - jQuery Mobile
  - Angular JS
  - Cordova/Phone Gap

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?

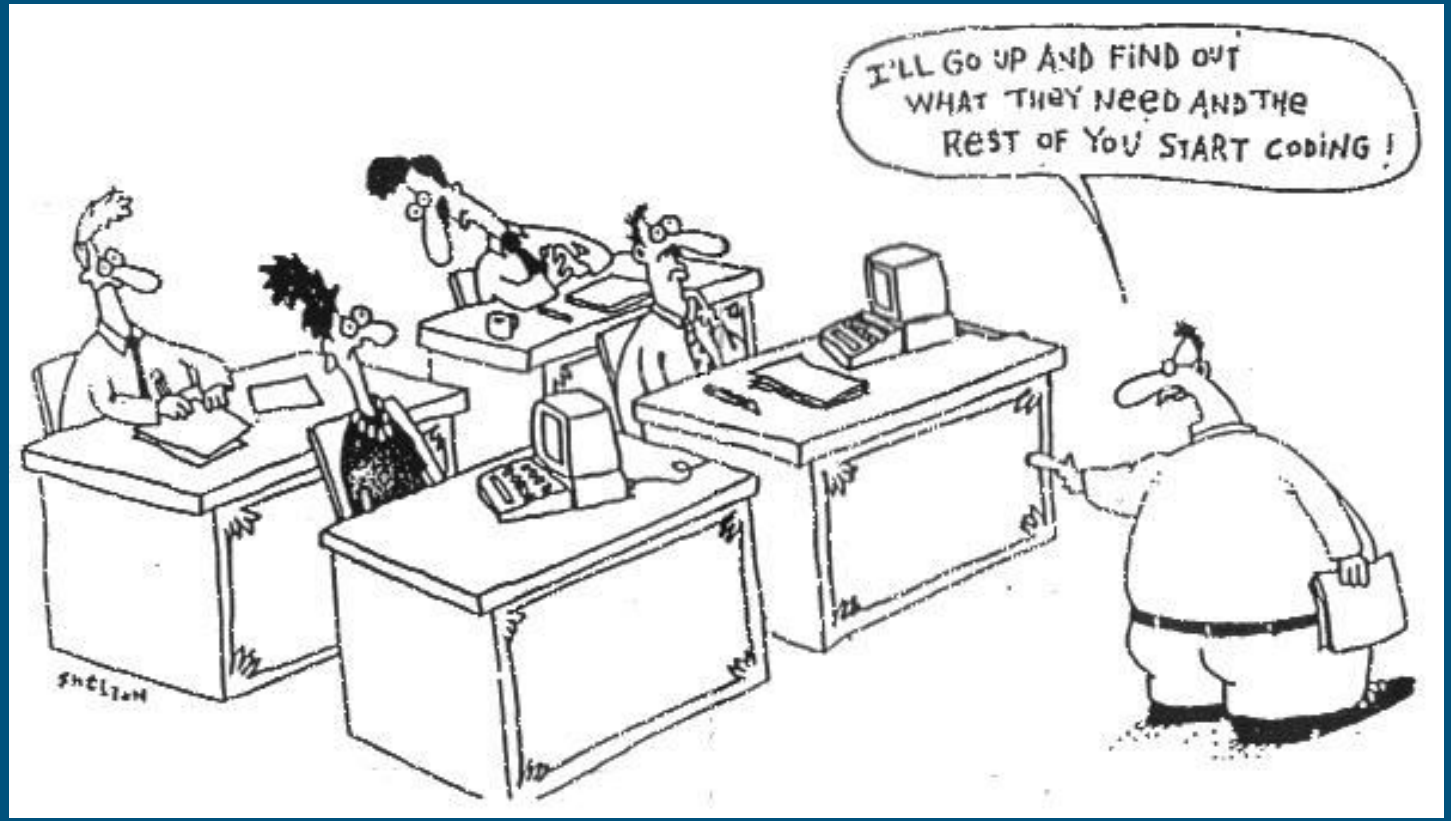
Programmier-Alltag 1 (U)

- ➔ 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; ...

Fragen


I'LL GO UP AND FIND OUT WHAT THEY NEED AND THE REST OF YOU START CODING!
SHELTON