TECHNISCHE UNIVERSITÄT DRESDEN

Praktikum Softwaretechnologie

Abschlusspräsentation

Gruppe 3

Fahrgastinformationssystem Eisenbahnlabor

Dresden, 02.02.2016

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Gliederung

Gruppe 3

Eric Schölzel
3. Semester
Diplom Informatik

Oliver Schmidt
3. Semester
Diplom Informatik

Robert Mörseburg
3. Semester
Diplom Informatik

Jonas Schenke
3. Semester
Bachelor Informatik

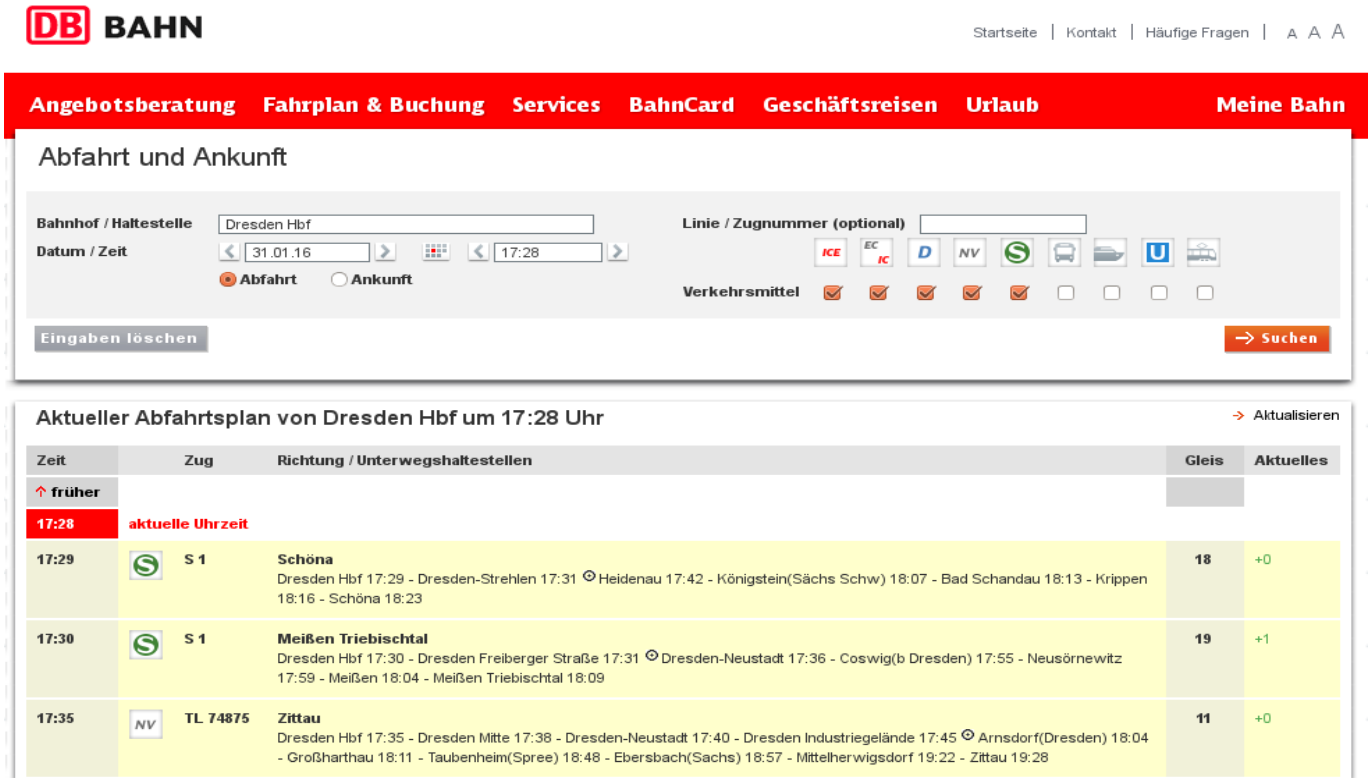
Zdravko Yanakiev
3. Semester
Bachelor Informatik

Betreuer

Dipl.-Medieninf. Ronny Kaiser
Fakultät Informatik
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie

- Webbasierte Ankunfts-, Abfahrts-, und Zuglaufanzeige
- Auswahl nach Bahnhof und Uhrzeit
- Optional Auswahl der Zuggattung
- Anzeige von Zwischenhalten
- Anzeige von Echtzeitinformationen (Verspätung, etc.) vom Fahrplanserver

- Orientierung: Online-Abfahrtsanzeige der Bahn



Muss-Kriterien

- Erreichbarkeit unter angegebener URL
- Kopfzeile (mit Logo, etc.)
- Fußzeile (mit Programmversion, etc.)
- Abfahrts- / Ankunfts- / Zuglaufanzeige
- Konfigurierbar
- Interaktiv (z.B. Anklicken eines Stops im Zuglauf)

Kann-Kriterien

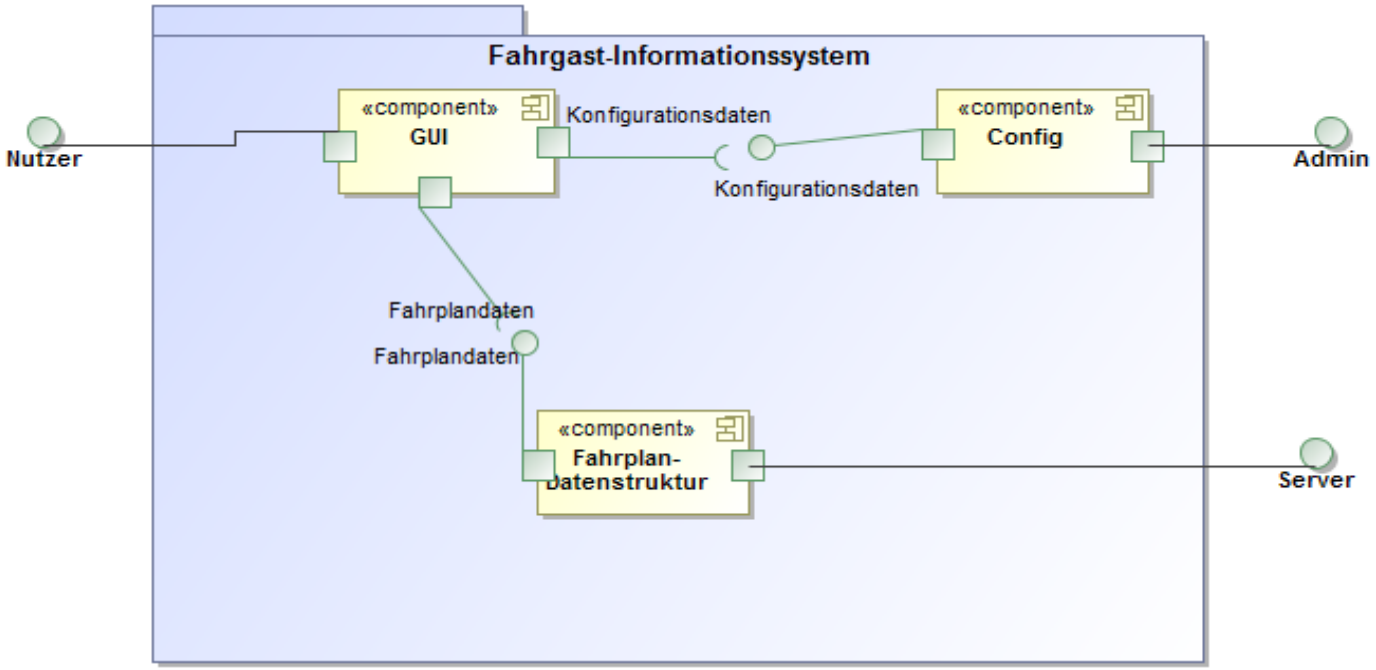
- Verbindungsstatus
- Zuglaufanzeige als Perlenschnur
- Ausblenden der Uhrzeit bei Verbindungsproblemen
- Vor- / Zurückfunktion im Browser verwendbar
- Logo konfigurierbar

Zusätzlicher Kundenwunsch

- Anzeige der Bahnhöfe auf interaktiver Karte
- Markieren des momentanen Bahnhofs

- GUI-Mockup
- Erstellung des Pflichtenheftes
- Erstellen eines Prototyps
- Planung mithilfe von UML-Diagrammen

Komponentendiagramm



Web

- Anzeige, Nutzerinteraktion, Filterung
- Entwurf mithilfe von Mockups
  - funktional
  - Nutzerfreundlich, optisch ansprechend

Datenstruktur

- Speichert Daten unabhängig von Datenquelle
- Wichtig: Datenintegrität
- Zentrales Bindeglied → Controller-Fassade

RailML®

- XML-Format zum Datenaustausch im Schienenverkehr
- Enthält Daten über die Infrastruktur, die Schienenfahrzeuge und den Fahrplan eines Eisenbahnsystems
- Der RailML-Parser stellt den anderen Programmkomponenten die Daten aus der RailML-Datei bereit
- Vorgehensweise: Trennung von XML-Parser und Datenstruktur → mehr Flexibilität

Telegramm-Struktur

- Entgegennehmen & Weiterreichen von Telegrammen
- Verschiedene Telegrammarten → Strategy
- Details während Entwurf unbekannt

- Java 1.8
- Spring Framework

Verlauf

1. Prototyp → nur kleine RailML
2. große RailML
3. Telegramme
4. Tests im Labor
5. Kundenwunsch

GUI

- Thymeleaf
- Datenweitergabe und Filterung im FisController

Datenstruktur

- Selbst unabhängig vom Spring-Framework
- TimetableController wertet Telegram-Objekte aus oder lädt Offline Fahrplan

RailML®-Parser

- Implementierung unter Benutzung des Spring Frameworks (Spring OXM)
- Laden der ganzen Datei in den Speicher
- Anschließende Verarbeitung in die interne Datenstruktur

Telegramm-Struktur

- Asynchrones Empfangen
- Parsen der Byte-Arrays → Little Endian, Spezifikation
- Synchrone Verarbeitung
- Modellierung der Telegram-Datentypen in Java
- Übergabe an Datenstruktur durch Events

Kundenwunsch

- Karte des Eisenbahnnetzes
- Realisiert mithilfe von HTML5/JavaScript (Canvas)

Probleme bei der Implementierung

- RailML nicht korrekt
- Telegrammspezifikation
- Telegrammteil sehr schwierig zu testen

- Aufgabenstellung erfüllt

- Enge Rückkopplung mit Kunden
- Kommunikation im Team
  - Git, GitHub
  - Wöchentliche Treffen
- Frameworks
  - Fokussierung
  - Hoher Einarbeitungsaufwand

6 Demonstration der Anwendung

Vielen Dank für Ihre Aufmerksamkeit!