Entwicklung einer unterstützenden Suchfunktion für den ConSol*-Profil-Generator

- Vorstellung
- Analyse
- Entwurf
- Implementierung
- Test
- Fazit

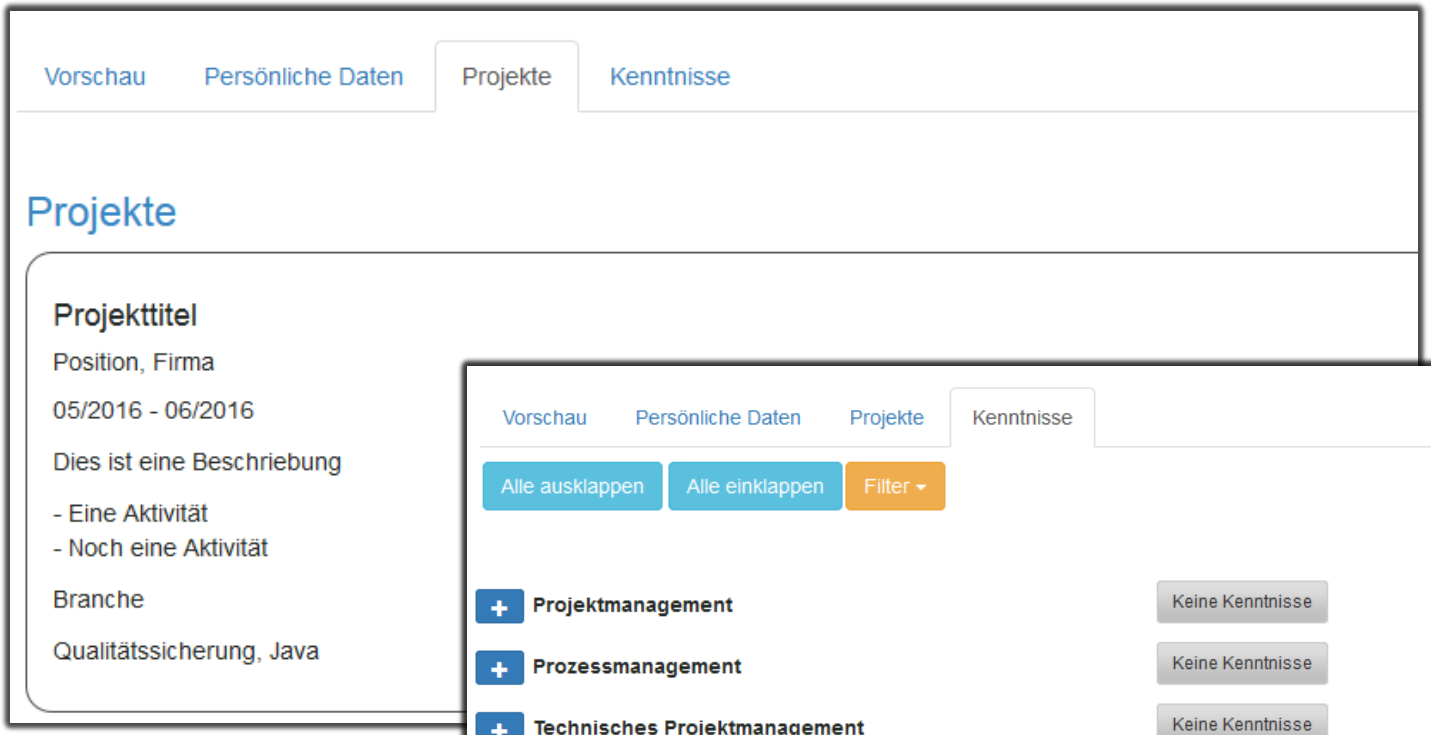
Agenda

consol
Wir unternehmen IT

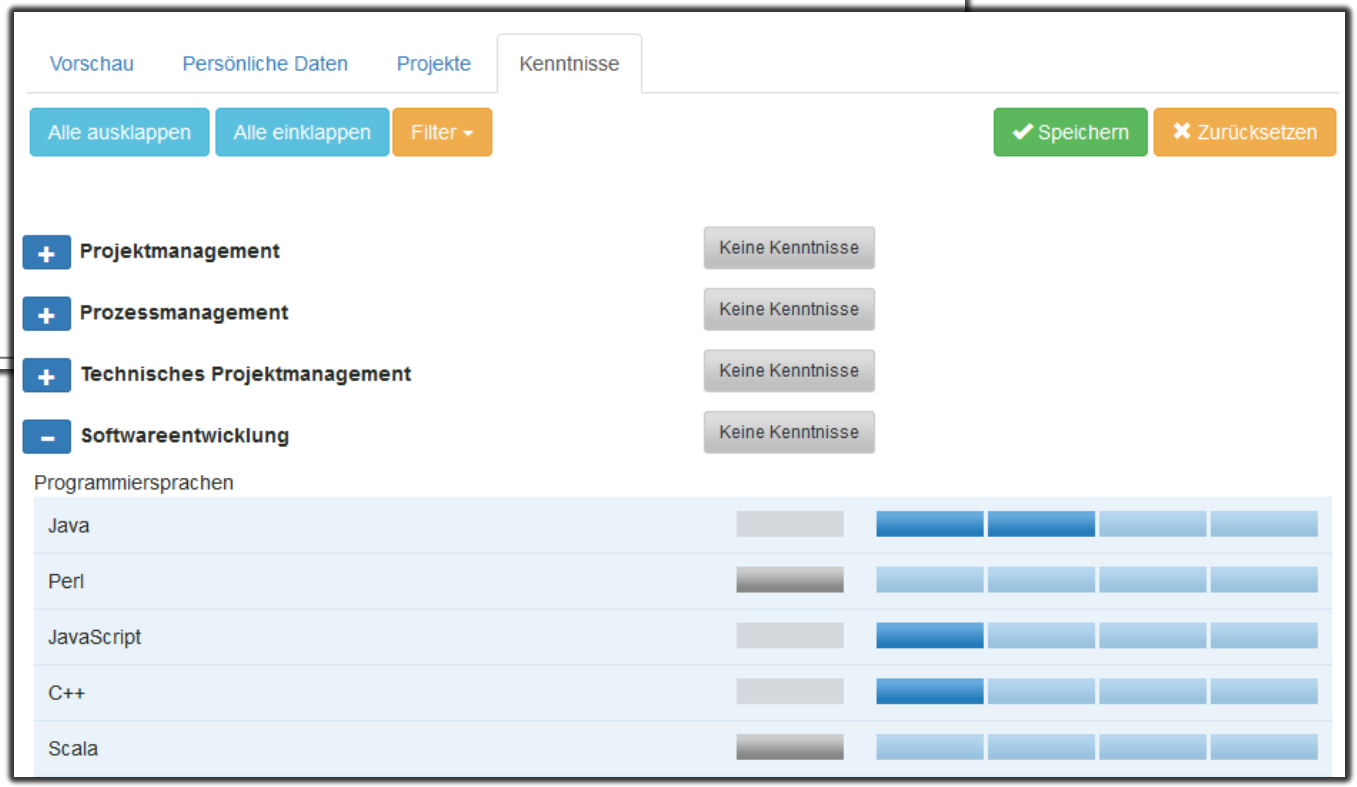
cm

Ausbildungsbetrieb

Analyse



Projekte



Kenntnisse

- Aktuell existiert keine Suchfunktion
- Das Finden geeigneter Mitarbeiter ist schwer
- Der Auswahlprozess wird manuell durchgeführt
- Der Suchprozess erfordert eine gute Intuition bzw. Vorkenntnis

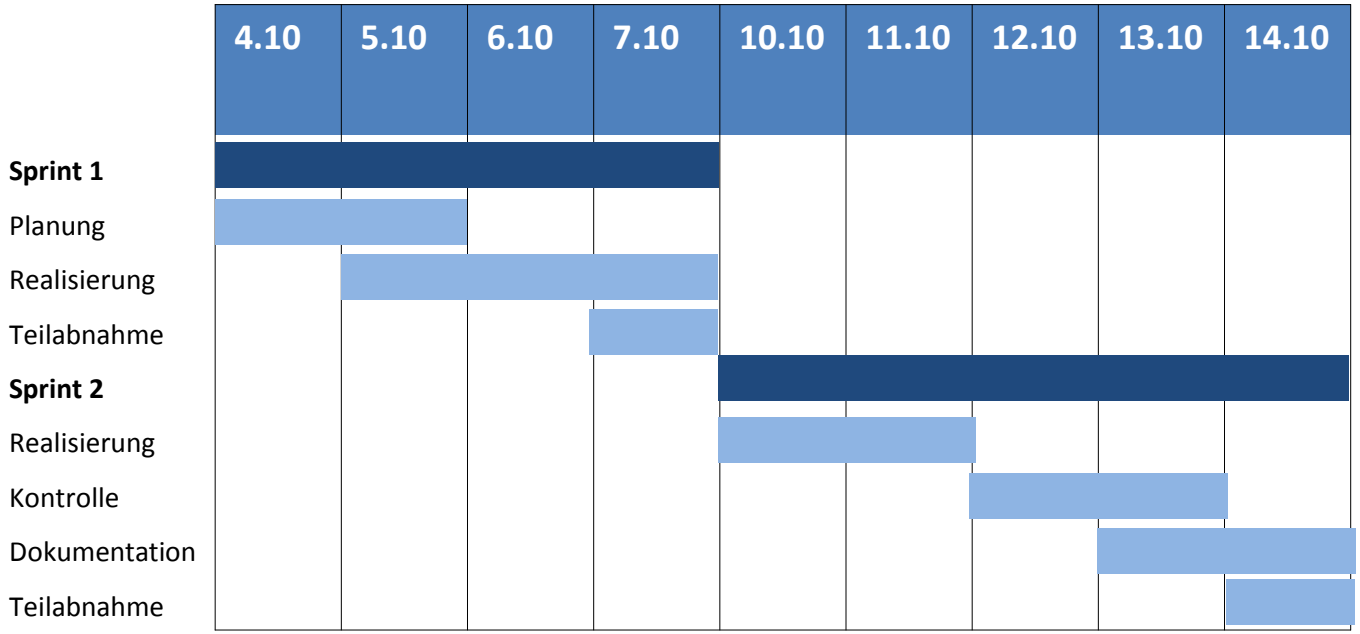
Projektbegründung

Entwicklung einer Suchfunktion:

- Unterstützung der Vertriebsabteilung
- Rationalisierung des Auswahlprozesses für ausgeschriebene Projekte
- Finden sachkundiger Kollegen (intern)

Projektziel

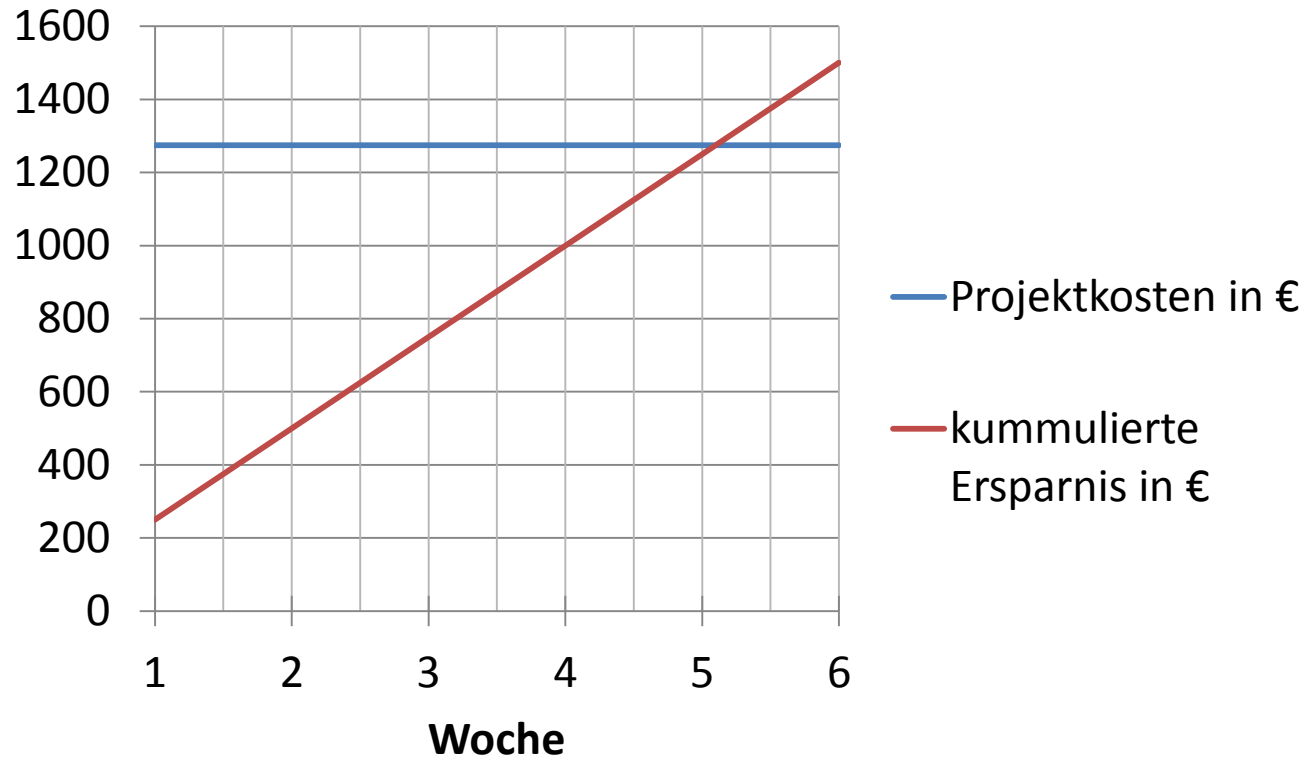
Scrum-Sprints



Entwicklungsprozess

Posten	Kosten
Arbeitszeit Entwickler	394,10 € (70 h)
Arbeitszeit Vertriebsmitarbeiter	180,00 € (6 h)
Gemeinkosten	700,00 €
Gesamtkosten	**1274,10 €**

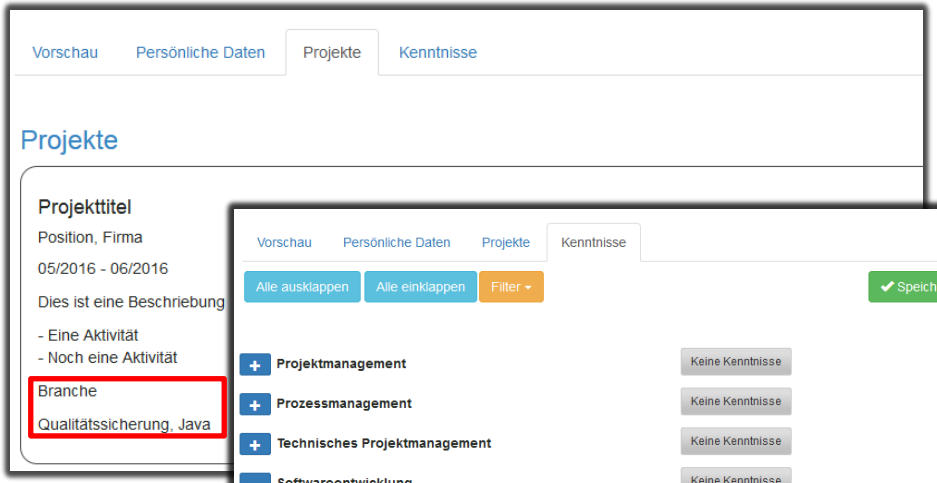
Projektkosten



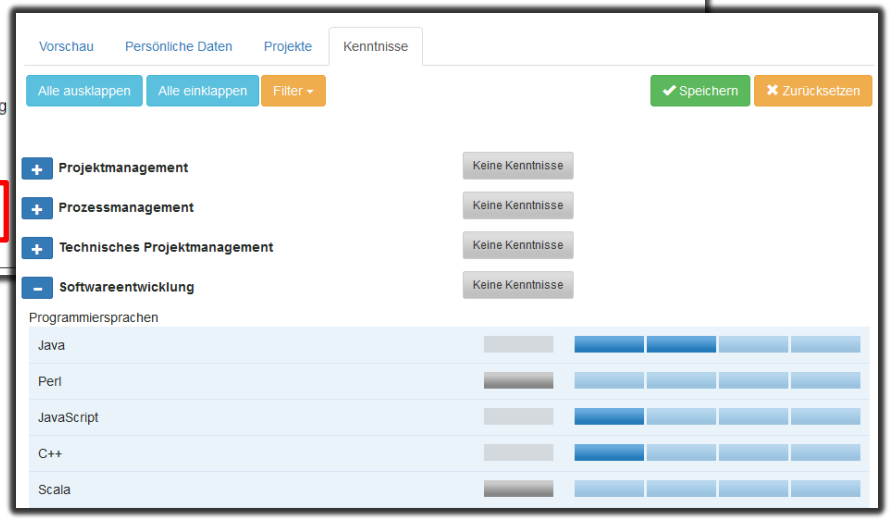
Armotisationsdauer

Durchsuchbarkeit von:

- Profilen
- Kenntnisse
- Branchen & Werkzeugen



Projekte

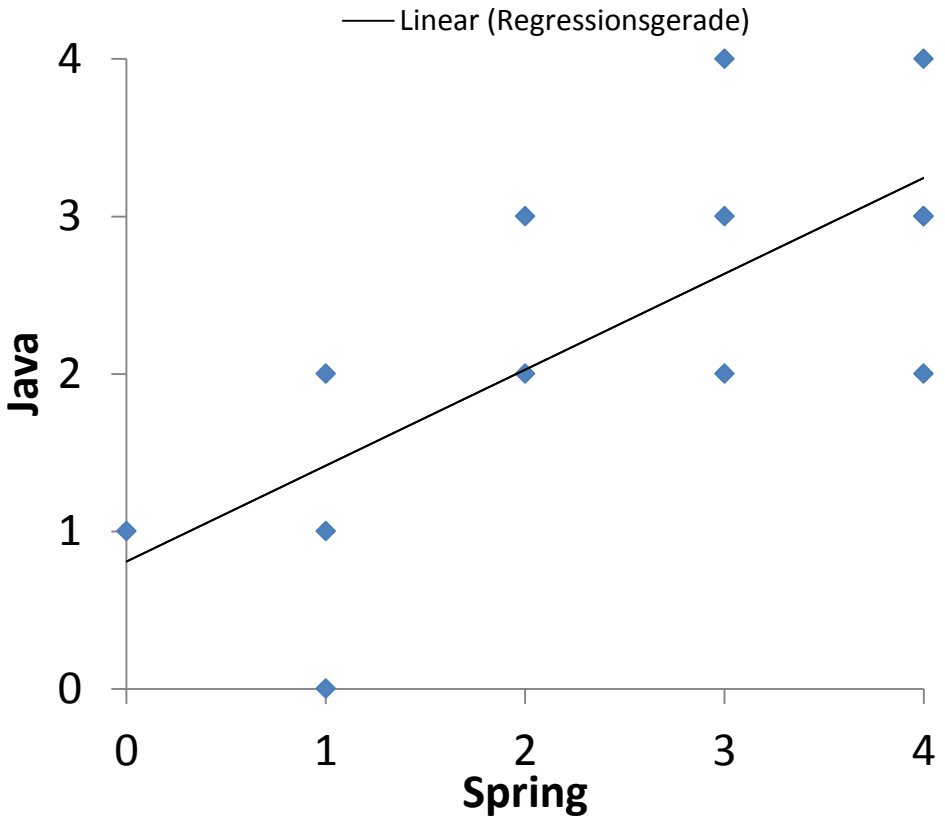


Kenntnisse

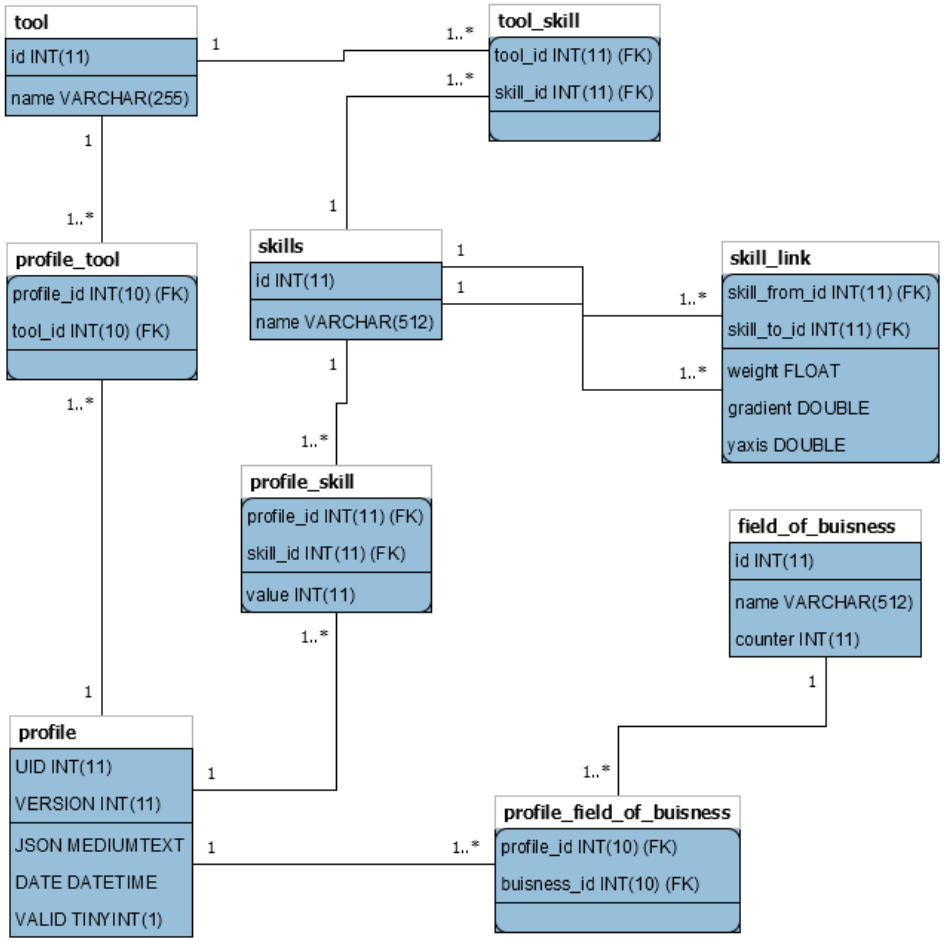
Soll-Konzept

Entwurf

- Korrelationen von Fähigkeiten
- Lineare Regression
- Freitextfelder im Projektdatensatz
- Fähigkeiten aus festem Wertebereich
- Abbildung von Werkzeugen auf Fähigkeiten

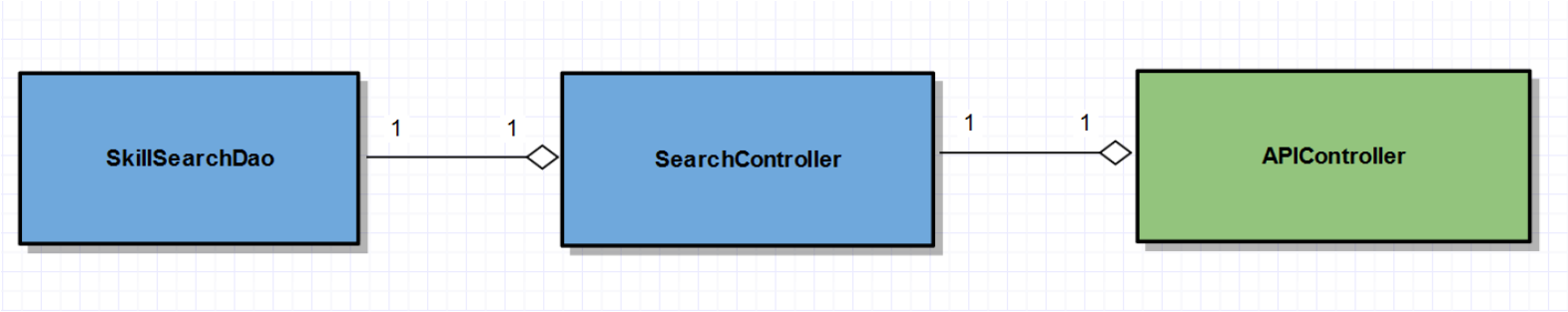


Risiken & Möglichkeiten



- Überführung des JSON-Modells in ein relationales Datenmodell
- Beschränkung auf relevante Felder
- Neue Datenstruktur für lineare Regressionsanalyse

Datenmodell



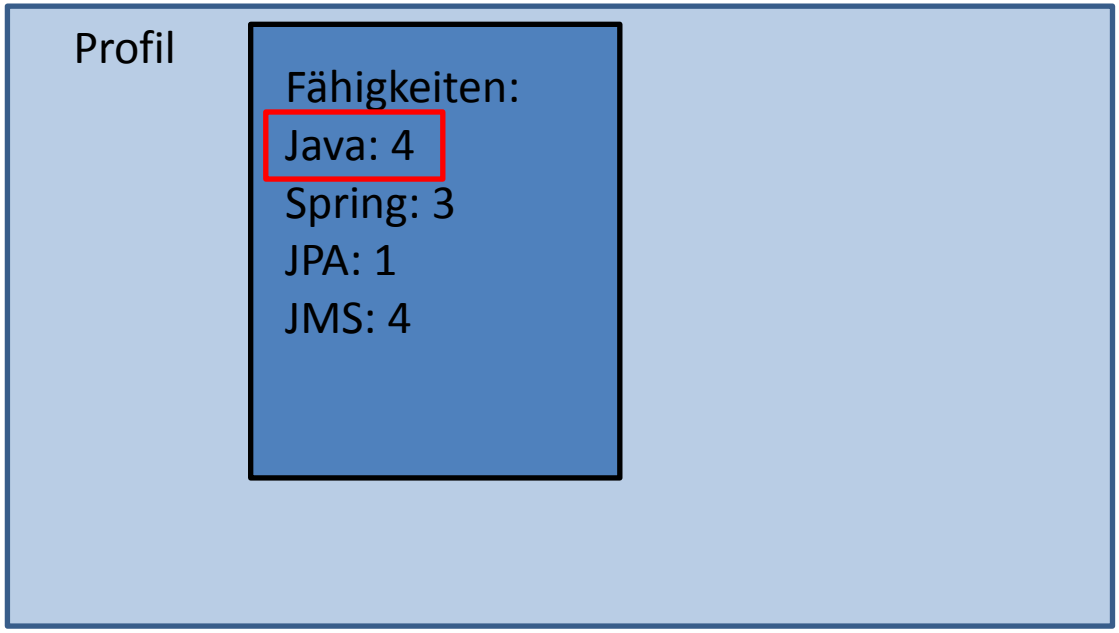
- API-Controller bereits vorhanden im Profil-Generator
- Zwei neue Spring-Beans (DAO, Controller)
- Schwache Kopplung, starke Kohäsion
- Tägliches Transformieren des JSON-Modells in relationales Modell

Geschäftslogik

- Eine Suche ist mittels Abfrage über das neue Datenbankschema einfach
- Standard: Suche über diskrete Menge an Kenntnissen
- Optional: Einbeziehung der Projekterfahrung

Scoringalgorithmus

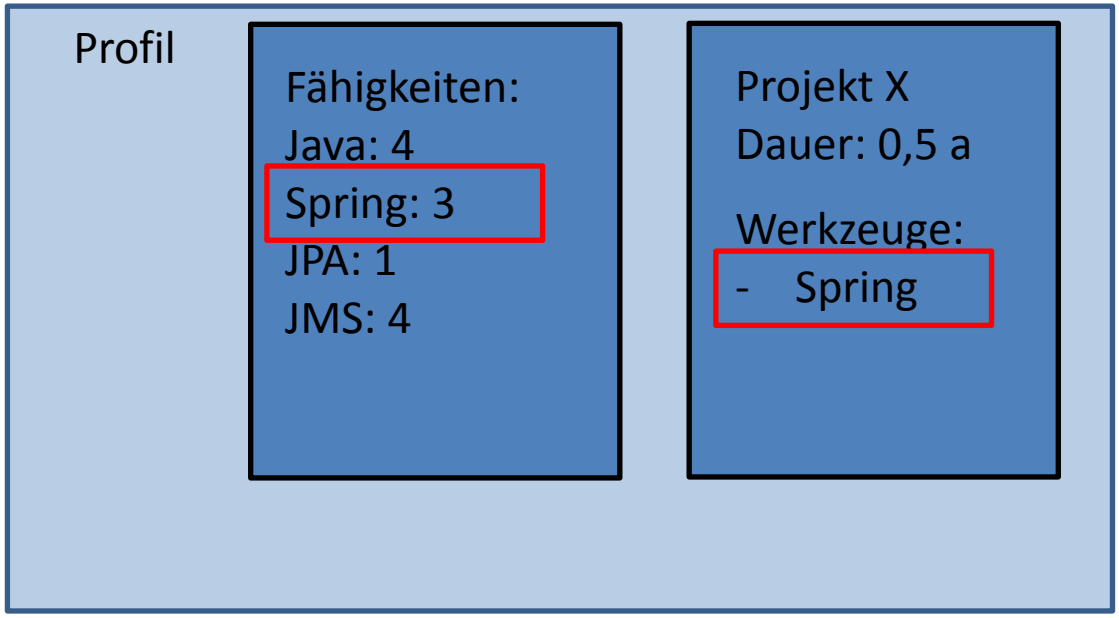
Suche nach Java (Berechnung für Fähigkeiten)

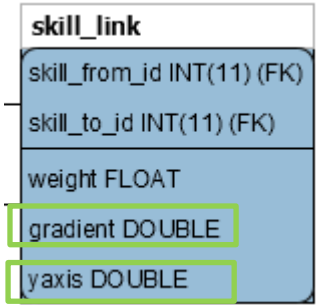


100 * 4 = 400

Scoringalgorithmus

Suche nach Java (Berechnung mit Werkzeugen aus Projekt X)





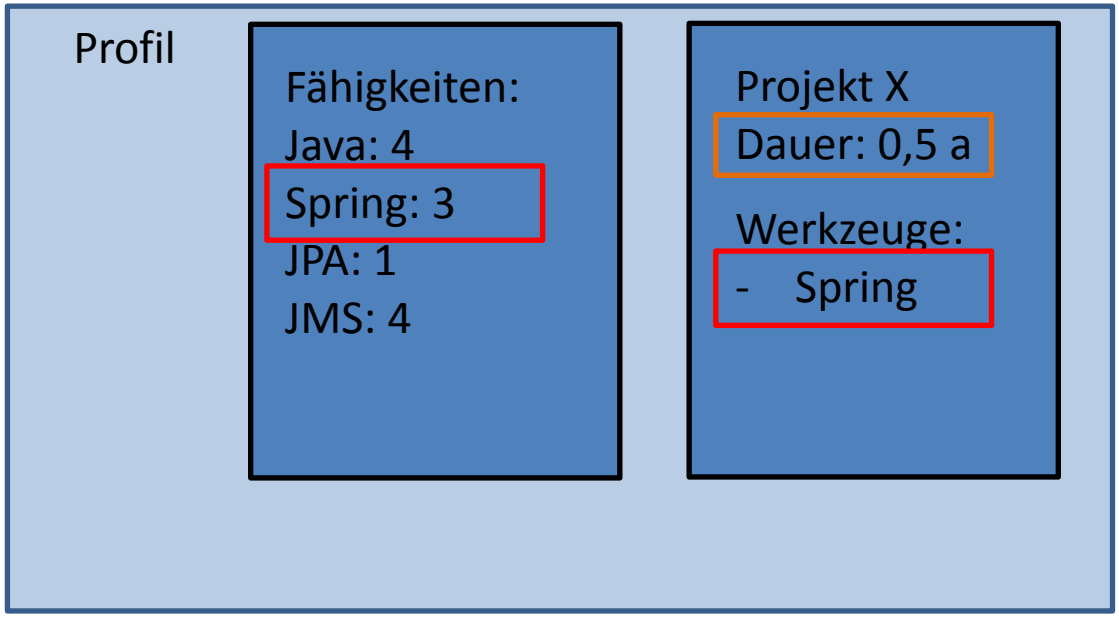
Werte der Regressionsanalyse

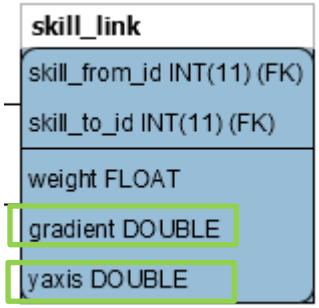
f(Java, Spring) := -0.61 + 3 * 0.94 = 2,247

Geradengleichung: y = mx + n

Scoringalgorithmus

Suche nach Java (Berechnung mit Werkzeugen aus Projekt X)





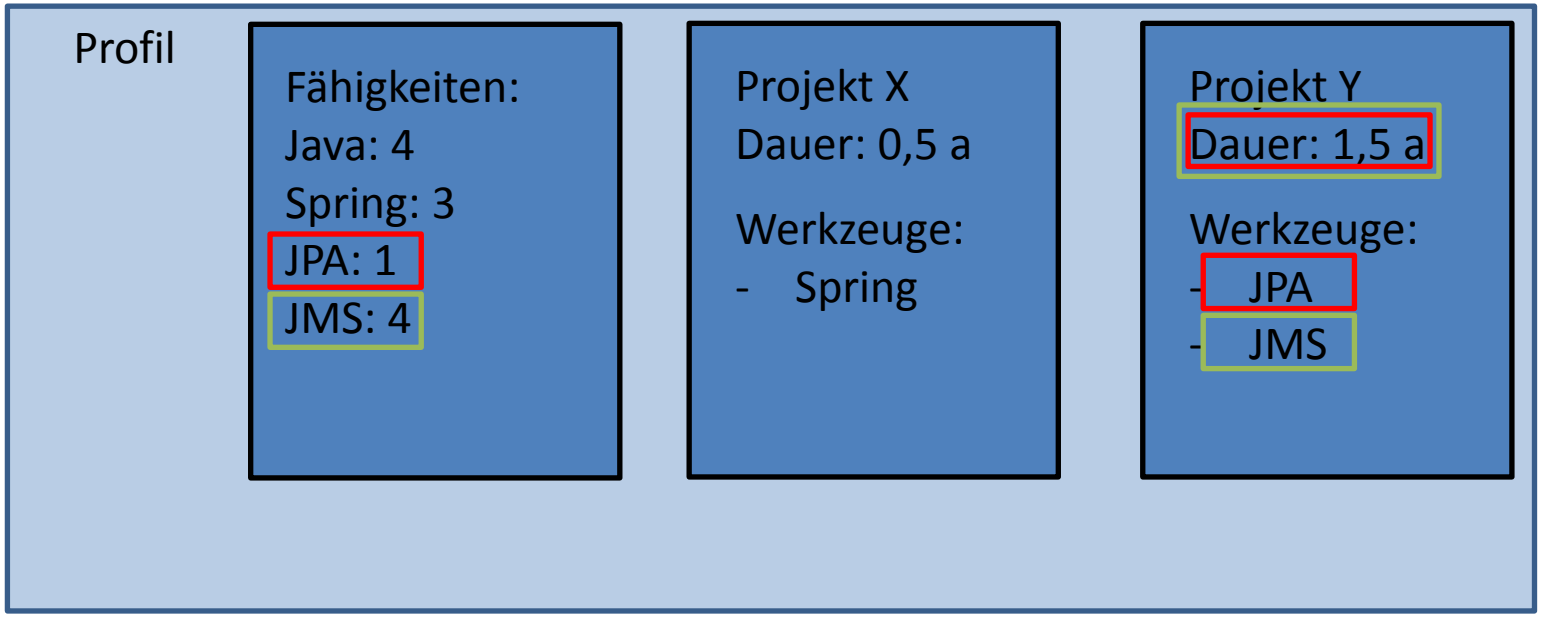
Werte der Regressionsanalyse

f(Java, Spring) := -0.61 + 3 * 0.94 = 2,247

400 + f(Java, Spring) * (182,5 / 365) = 401,123

Scoringalgorithmus

Suche nach Java:



401,123 + f(Java, JPA) * 1,5 + f(Java, JMS) * 1,5 = 401,123 + 0,435 + 3,822 = 405,38

Scoringalgorithmus

$$100 \cdot x_b + \sum_{i=1}^{NumProjekte} \left(\left[\sum_{j=1}^{NumWerkzeuge(Projekt(i))} f(x, y_j) \cdot \frac{Projektdauer}{365} \right] + b(p_i) \right)$$

- x: gesuchte Fähigkeit
- x_b: Bewertung der Fähigkeit im Profil des Mitarbeiters
- y: dem Werkzeug zugeordnete Fähigkeit
- f(x, y): Regressionswert von x auf y, wenn Korrelationskoeffizient >= 0.85, sonst 0
- b(p): Branchenscoring
- i: Index der Projekte im Profil
- j: Index der Werkzeuge im Projekt

Scoringalgorithmus

Implementierung

- Datenbank-Setup mittels MySQL-Workbench
- Datenbankzugriff mittels Spring-DAO-Bean (JDBC-Template)
- Spring-Controller-Bean
- Context and Dependency Injection (CDI)
- Spring Annotationen
- Frontend mittels Backbone.js

```
public class SkillSearchDao {

    @Resource(name = "profgenDatasource")
    DataSource profgenDatasource;
    private JdbcTemplate jdbcTemplate;
    @Autowired
    private ProfileDao profileDao;

    @PostConstruct
    private void initDs() { this.jdbcTemplate = new JdbcTempla
```

Realisierung

Test

- JUnit als Testframework
- Spring Boot bietet eine in-memory H2-Datenbank an
- Phantom.js als Test-Client für Integrationstests
- Direkter Zugriff auf Datenbank um DAO-Funktionalität zu testen

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = IntegrationTestContext.class)
@WebAppConfiguration
@ActiveProfiles("test")
@IntegrationTest("localhost:8080")

/**
 * All tests depend on the data inserted via the data.sql file. We test against a in memory db.
 */
public class SearchControllerTests {
```

Test-Kontext

Fazit

- Erfüllung des Soll-Konzepts: Relevante Felder und Fähigkeiten eines Profils sind durchsuchbar
- Zeitliche und inhaltliche Planung wurden eingehalten
- Der Scoring-Algorithmus liefert wertvolle Ergebnisse

Ausblick:

- Der Algorithmus kann ggf. nach Nutzerfeedback verfeinert werden
- Überführung in den produktiven Betrieb

Fazit

Vielen Dank für Ihre Aufmerksamkeit