Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelor's Thesis

A Recommender Framework for Skills Management

In Cooperation with SinnerSchrader

Torben Reetz

3reetz@informatik.uni-hamburg.de
B.Sc. Software Systems Development
Matr.-Nr. 6524064
Semester 07

Primary Supervisor: Prof. Dr. Walid Maalej
Secondary Supervisor: Prof. Dr. Eva Bittner

February 2017

Abstract

Nowadays, working environments do not present workers with a continuous stream of recurring tasks like at the assembly line. Instead, highly skilled employees with diverse competencies work in dynamic teams to complete nonrepeating projects. As a consequence, project driven organizations face the problem of constantly needing to put teams together based on the members' skills, experience, motivation and personal preferences. At SinnerSchrader, like in many businesses, there is no central source of information about this data. This thesis covers the design and partial implementation of a web application custom-tailored to SinnerSchrader that provides a simple way to manage employees' skills and interests using a novel approach of integrating a rating based Information Retrieval System ranking employees not only by their knowledge but also by their motivation to find the most suitable employee for the user's search query. To validate that the application, and especially the novel scoring approach, is capable of fulfilling the users' needs, a survey has been conducted and interpreted. The backend implementation has been tested regarding both functional and non-functional requirements.

Contents

1. Introduction

1.1. Motivation

Project driven organizations face the problem of constantly needing to put teams together based on the members' skills, experience and preferences. In many businesses, there is no adequate source of information about those data which makes finding the right person with a specific ability even more complicated. A popular approach to this problem is using computer programs to find an employee skilled in a given set of tasks.

SinnerSchrader, a Hamburg based web agency that will serve as the practical context for this thesis, decided to launch an internal application for skills management that is meant to solve the aforementioned problems. This thesis will deal with the design, concept, partial implementation and evaluation of said application.

1.1.1. SinnerSchrader

The SinnerSchrader Group is a full-service web agency based in Hamburg aggregating the subcompanies SinnerSchrader Deutschland, SinnerSchrader Content, SinnerSchrader Commerce and SinnerSchrader Swipe. The broad spectrum of expertise, including, but not limited to, digital communication strategies, visual and interaction design, technical architecture, full stack development, editorial services, content production, e-commerce, mobile app development, hosting, and maintenance, allows SinnerSchrader to serve all needs regarding their customers' digital transformation. The combination of all said competencies under one single roof reduces organizational friction between the discipline-specific teams, because they all share the same vision of the big picture they are creating. This does not only lead to faster development cycles but also to a more coherent and unified product. If not stated otherwise, the name *SinnerSchrader* will further refer to SinnerSchrader Deutschland.

Project-Driven Business

Being a web agency, SinnerSchrader operates in a project-driven way. This means there is no continuous stream of recurring work repeating constantly, but many different projects for different clients, each one dealing with varying challenges and questions. From a technical point of view, know-how needed for each project is extremely diverse since every application uses its own dedicated stack of technologies. As a consequence, the developers' skill sets are based on the combination of projects they have worked on and their general field of interest. This results in managers frequently having to put teams together based on the members' skills with respect to the individual requirements of the project.

Matrix organization

The personnel of SinnerSchrader is divided into two different types of teams: functional teams of employees sharing the same specialization, e.g. backend development, frontend development, design, business strategy, or concept, and project teams of people from different functional teams working collaboratively on the same project. This structural model is called a *matrix organization* [Ber16, P. 75]. The organizational head of functional teams will further be called the *supervisor*; the counterpart for project teams will be called *team manager*.

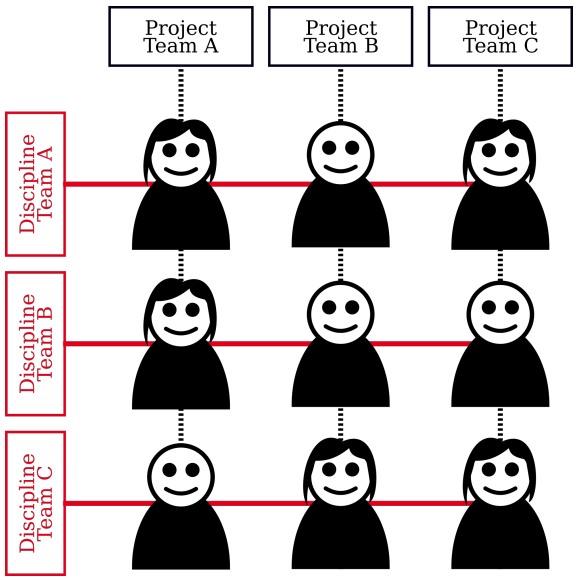

Figure 1.1.: Illustration of a matrix organization

1.2. Leading Goals

The main goal of this thesis is to design, partially implement, and evaluate a skills management application custom-tailored to SinnerSchrader using a novel approach to search for employees. Instead of returning the most skilled persons, the tool will list the most suitable ones, employing a ranking mechanism that measures how well someone fits into the searched skill set. An analysis of the user's expectations will be used to define *suitability* in this context. Furthermore, recommender systems that enrich the user experience will be evaluated, designed, and implemented. The created methods and implementations will be evaluated both technically and conceptually.

2. Related Work

Skills management is a trending topic in today's management world, as it is a vital part of the success of a modern business. Darvish et al. highlight the importance of knowledge management in various forms of institutions and compare different knowledge management strategies. Moreover, the authors show how enterprises can introduce tools and mindsets to use the positive effects of knowledge management in their organizations [DAQ13], but do not discuss concrete software tools. A market analysis by Lehner, however, compares multiple commercially available software systems for skills management and provides information adapted in 3.5 [Leh04].
Beck outlines a case study that deals with the introduction of a skills management system at *Putzmeister GmbH* and reveals important success factors of such a software, e.g. legal concerns and cooperation with the works council, obstacles that occur in the maintenance phase of the system's lifecycle, usability requirements, and ways to motivate employees to provide a sufficient amount of data. [Bec03].
In contrast to the systems analyzed by Lehner and Beck, the application that this thesis will deal with does not only show the information saved in its database, but also provides a powerful search function, and recommends employees based on the combination of multiple personal factors including their motivation.
The concept of approaching team building and management challenges with algorithms has been evaluated and successfully implemented multiple times. In 2013, Ivanovksa et al. compared various data mining algorithms for the automatic composition of teams and propose the usage of *Bayesian Networks* for this kind of problems [IIK13]. Unfortunately, the authors do not take into account factors like the employees' motiviation and satisfaction. Those aspects have been examined by Canós-Darós who introduced an algorithm to measure employees' motivation [Can13] and highlights qualitative factors the application should implement. Spoonamore et al. created an algorithm that deals with the matching of personnel to open positions in the *United States Navy* [SSH07]. An adaption of this method lays the foundation for an algorithm used in this application that scores how well an employee fits into a certain skill set (see 4.1.3). Furthermore, the authors describe non-technical requirements an algorithm which ranks personal abilities has to meet in order to be accepted by the target audience that will be scored by it.
This thesis does not cover the visual concept and implementation of the application's graphical user interface since Strecker's bachelor's thesis addresses this field of functionality [Str17].

3. Requirements and Existing Solutions

3.1. Usage Scenarios

3.1.1. Asking for Help

Having the possibility to ask for help is a vital part of the working culture in many knowledge driven businesses, so collaboration and the sharing of ideas are a major factor in the company guidelines of SinnerSchrader. The application is supposed to act as a central repository for knowledge and contacts, enabling employees to find someone who can help in order to answer questions and find solutions to domain-specific problems.

3.1.2. Finding Potential Team Members

Team managers constantly face the problem of reassembling parts of their teams, forming new teams for new projects, and disbanding teams whose projects have ended. As there is no unified source of information about all employees at SinnerSchrader, managers often do not find the most suitable team member to fill an open position because they simply do not know each other yet. This tool will give managers the opportunity to search the entirety of SinnerSchrader's workforce and find the most suitable one meeting all requirements of the open position, thus making collocating teams easier and more efficient.

3.2. Collecting the Required Featues

As shown in 3.1, the application is supposed to provide help for two target audiences: project managers staffing their teams, and employees looking for help with a certain problem. The actual features the system should include have been collected by the *Flow Team*, a group of project managers who coordinate and monitor project team staffing and personnel development. This team also supervises the development of the skills management application. To validate that the requirements asked by them reflect the end users' needs, and to obtain a better impression thereof, semi-structured interviews have been conducted. In addition to collecting the asked persons' general expectations, the interviews aimed to give answers to these concrete questions:

- What should be the range of the scale used to measure knowledge and motivation?
- Should the overall experience of the employees be captured by the system?
- Should the system differentiate between short-term and long-term motivation?
- Should users be able to rate their colleagues' skills?
- Should the system be able to automatically form project teams?

3.2.1. Interview Procedure

The interviews consist of a ten-minute-long[1] open discussion between the interviewer and the interviewees, so that in the event of the interviewee giving an extraordinaire answer or bringing up a new idea, the interviewer may ask for further explanations. The questions listed in 3.2 will provide a guideline to structure the interviews. Before the interview, the interviewees have been informed about the procedure, the personal data that will be collected for the interview and the purpose of its collection, and had to give their written consent (see B.1). In order to extract all information from the interviewees' answers, audio recordings have been taped during the interviews.

[1] The actual timeframe may vary depending on the interviewee's answers.

3.2.2. Interviewees

As described, there are three main groups of stakeholders regarding the application: the Flow Team that supervises its development, project managers who search for new team members, and *regular* employees looking for help. For each of these groups, one representative has been interviewed:

- Ms. Spranger
  Ms. Spranger is the Product Owner in the developent team and part of the Flow Team for which she will speak.

- Mr. Warnholz
  Mr. Warnholz is a project manager and has many years of experience with various teams at SinnerSchrader. He frequently searches for employees fitting into specific project teams and will represent the project managers' point of view.

- Mr. Gruber
  Mr. Gruber is a backend developer and supervisor of one of SinnerSchrader's domain-specific teams. As he worked on multiple projects that use drastically different technology stacks and processes, he is familiar with the problem of having to look for colleagues that can help with a specific problem.

In contrast to Ms. Spranger, Mr. Warnholz and Mr. Gruber have not been in contact with the development of the skills management application, so that their opinions should not be influenced by internal decisions made about the tool's design.

3.2.3. Results

The interview recordings have been transcribed (see B.2) to extract information about the interviewees' opinions to decide whether or not the respective feature should be included in the application, and if so, how it should be designed.

General Expectations

Popular features requested by the interviewees include the ability to see how motivated employees are regarding specific skills in order to increase employee satisfaction and to deduce a certain employee's development of interest. Furthermore, one of the most desired features is that project managers get a tool to find suitable members for their teams. Qualitative factors the interviewees requested are that the tool is easy to use and that the interface is reduced to the most important components.

Scale

When asked for a rough estimate of how the scale on which knowledge and motivation will be measured should be designed, all interviewees replied that they prefer a simple scale between four and six values. Mr. Gruber and Mr. Warnholz proposed to use five step *Likert-Style Items*. Since those items are based on the idea of estimating the distance to a neutral element in both directions (e.g. better or worse), this type of scale is not applicable for the measurement of knowledge. The chosen approach will include a scale of four items for both types of values:

Knowledge	Motivation
novice	uninterested
basic knowledge	indifferent
advanced knowledge	somewhat interested
expert	highly interested

Overall Experience

All interviewees agree that the overall experience of the employees should not be included in the search function because the skill-specific experience is significantly more meaningful in this context. The general grade of experience that is reflected by the employee's title (e.g. *Junior*, *Intermediate* or *Senior*) should be shown in the result list but it must not influence the search or rating.

Duration of Motivation

Regarding the question whether the duration of a will, that is how long the interest in a specific skill will persist, should be captured in the system, the interviewees agree that this ought not be the case. The system is supposed to present a snapshot of the present situation and should not be used for long term assumptions. Another important factor stated by the interviewees is that forecasts about one's motivation will be highly inaccurate.

Peer Rating

All of the interviewees answered that they do not want employees to be able to rate each other's skills because they fear that subjective impressions might blur the professional view leading to a heavy influence of sympathy on the ratings. Furthermore, the works council prohibits the introduction of such a feature (see 4.4) because multiple employees raised concerns about this kind of functionality and wish it not to be included. To respect their demands, users of the application will not be able to rate someone other than themselves.

Generating Teams

The application will not save which projects each employee works on or has previously worked on, because this information is managed by another internal service which will not be replaced by this application. This data would then have to be updated in both systems manually. Under this condition, no interviewee wants the application to automatically compose project teams. Reasons that have been mentioned are that maintaining the same data in two systems[2] is impractical, that this task is too complex to be automated, and that composing complete teams is not a popular use case for it. Project teams at SinnerSchrader grow organically, so it is much more common for project managers to look for one single potential member to add to their team instead of composing a completely new one. Based on these insights, the application will not be able to compose teams.

3.3. Workflows for Collecting Data

The application will give employees the opportunity to provide information about their personal knowledge regarding their skills. Furthermore, they can assign a will value for every skill that describes if they prefer doing the implicitly linked activity or working with the tool described by said skill. That is, people can define what they want to do and what tasks they would like to refuse.
With employees continually enlarging their knowledge and their fields of interest shifting towards new technologies, tools, or even functional divisions, providing data about their skills and preference only once will lead to the system being filled with obsolete information. The quality of the search results and suggestions heavily relies on the fidelity and volume of the underlying data about the employees, hence keeping that information up to date is crucial to the performance of the application.

3.3.1. Biannual Feedback Meetings

Every employee has biannual meetings with their supervisor to exchange bidirectional feedback, define personal goals, and negotiate possible changes of salary. Part of the feedback given by the employee are subjects they learned or enhanced their knowledge about, and newly developed interests. These insights are documented and registered in the employee's personnel file. The aforementioned meeting will be the regular occasion for supervisors and employees to refine the data saved in the application and to add newly gained skills to it. The supervisor is advised to address discrepancies between the employee's estimations of skills and their own one to accommodate the human factors of self-perception. A case study performed by Beck indicates that this approach to collect information about the employees' skills generates a sufficient amount of data to run such a system [Bec03].

[2] The project management system and the skills management application

3.3.2. Data Maintenance

Employees and supervisors are encouraged to be in rich contact with each other in order to deliver continuous feedback about the individual person's needs, impediments, and the status of their current projects. As a result, supervisors can identify appropriate moments for reevaluating the skills and preferences saved in the application and notify the employee.
For example, according to Tuckman's team development model, the so called *adjourning* phase of a project is an occasion for "recognition of individual achievements and reflection on how far the team has come" [Wil10, P. 3] and thus is a convenient chance to add new skills acquired during the project and to refine the existing data.
In contrast to the supervisor, the application will not be able to find the best situations to notify employees to maintain their skill profile, so sending automatic notifications will not be nearly as effective as being reminded by the supervisor. Furthermore, according to the *Direct Marketing Association (UK)*[3], only about one in five automatically generated e-mails will be opened [DMA14], which reflects SinnerSchrader's experiences with email reminders. Those facts justify the decision that the system will not send any notification to its users.

Intrinsic Motivations

In addition to the mentioned reasons for employees to provide data about their skills which are all extrinsic motivations, there also are intrinsic motivations to do so. Being motivated intrinsically is defined as "doing something because it is inherently interesting or enjoyable" [RD00]. As people are motivated to focus on tasks they fancy, they are also motivated to voluntarily keep an eye on the quality of the data that is used to determine the tasks they will have to perform.

[3] https://dma.org.uk/

3.4. Requirements

Based on the Flow Team's demands and the data collected in the interviews, requirements for to application have been composed. The functional requirements describe features which will be provided by the system, whereas non-functional requirements define framework conditions for its performance.

3.4.1. Functional Requirements

- Accessible to all Employees
  Every employee must be able to use the application regardless of their equipment or preferred operating system.
- User Profiles
  Anyone can see another user's profile consisting of basic information about the user such as name, location, e-mail and personal skills. Personal skills are composed of a name, a skill level, and a will level. The last two describe the employee's knowledge and motivation regarding the respective skill.
- Provide/Edit skills
  Users can add new skills from a pool of predefined skills to their own profile. Already added skills can be edited and removed from the profile.
- Search
  A search function can be used to find people who have added one or more specific skills to their profile. When searching for multiple skills, only persons matching all of them will be displayed.
  - Ranking
    By default, the search results' order will be defined by a score aggregating the individual employee's skill level, will level and grade of specialization in the searched skills.
  - Sorting
    The users are able to sort the search results not only by said score but also by skill and will level.
  - Provide Additional Information
    Employees are able to add comments to their profile to give extra information about working hours, special contracts, certificates they own, etc.
- Management of Known Skills
  New skills can be added to the set of known skills in the application. Existing skills can be edited and removed. Users' personal skills are automatically updated when a skill has been edited, so that the integrity of the users' profiles is maintained at all times.

3.4.2. Non-Functional Requirements

- Device Types
  Even tough smartphones are gaining a constantly increasing share in terms of total internet traffic [Sta], the application will be optimized for desktop use. Every employee has permanent access to their computer and uses it for their work, so it can be assumed that the very same machines will be used to access the skill management application.

- Browsers
  Every employee is allowed and encouraged to install their favorite software on their personal computer, so nearly every web browser can be found. The application should run on *Google Chrome*[4], *Mozilla Firefox*[5] and *Safari*[6] in their latest versions. *Internet Explorer*[7] and *Edge*[8] will not be supported. Explanations on these decisions are given by Strecker [Str17].

- Scalability
  SinnerSchrader has 459 full-time employees [Sin17], so the application will be designed for approximately 500 users. In the event of rapid growth in the userbase, e.g. due to the opening of another office, the system will have to scale up to handle the larger number of users.

- Load/Response Times
  According to *Google's* RAIL model, a website needs to respond to the user's input within 100ms to offer a fluent user experience; the triggered action should be finished within one second after the user's interaction [Kea17]. In the context of the application's search function, this means that the system will show the user that their input has been acknowledged within 100ms. Within one second after the submission of the search request, the result list will be rendered completely.

3.5. Commercial Solutions

Three commercial skills management applications have been picked randomly for further examination: *Skills Base*, *Talent Management (engage!)* and *SkillsDB Pro*. A more detailed analysis by Lehner shows that most tools provide a spectrum of features and limitations very similar to the examined solutions [Leh04], so that the selection is assumed to be representative to the market.

[4] https://www.google.com/chrome/browser/desktop/index.html
[5] https://www.mozilla.org/en-US/firefox/products/
[6] http://www.apple.com/lae/safari/
[7] https://www.microsoft.com/en-us/download/internet-explorer.aspx
[8] https://www.microsoft.com/en-us/windows/microsoft-edge

3.5.1. Skills Base

Skills Base[9] offers most of the required features, but also includes a large number of functionality SinnerSchrader does not need and is not willing to use. This includes assessments, the categorization of skills, and a role model for advanced access rights configuration. A central aspect of the application are dashboards displaying information about the most popular skills in the organization and long term statistics. The API supports searching for multiple skills with minimum levels in knowledge and interest. Unfortunately, the application cannot be hosted by SinnerSchrader. This means the employees data would have to be stored on external infrastructure that is operated and monitored by an external company, which should be avoided due to legal reasons.

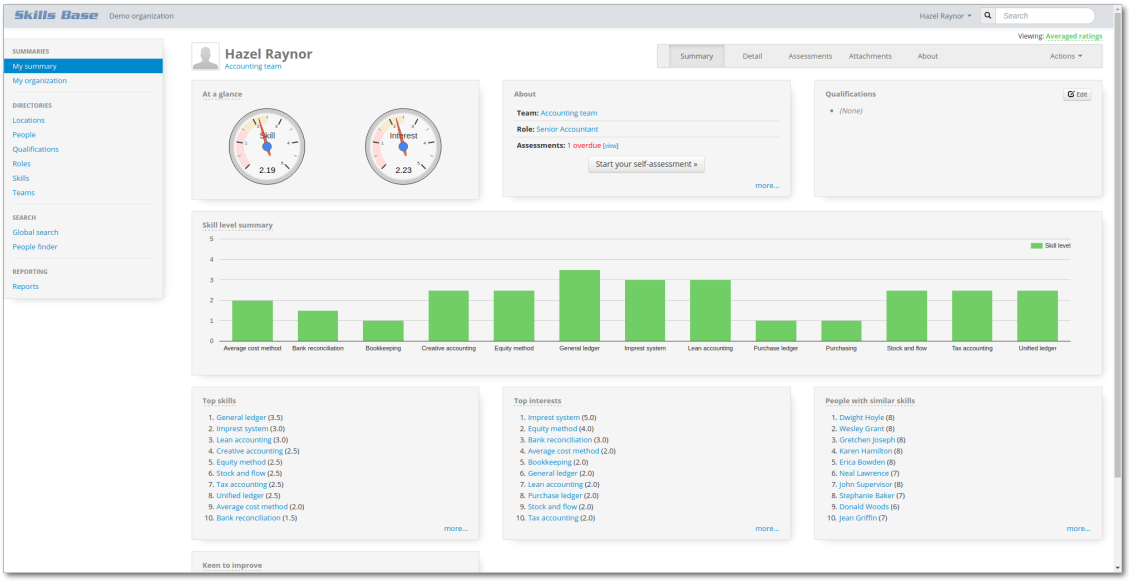

Figure 3.1.: SkillsBase Dashboard (source: *https://www.youtube.com/user/SkillsBase*)

3.5.2. Talent Management (engage!)

Talent Management[10] is a module for *Infoniqa's* management software engage![11]. It offers advanced features for managers such as a powerful search function controlled via a special query language. It also includes data about the employees' salaries, feedback protocols, and certificates, but lacks the feature to register motivation. It can only be used in combination with engage!, a complete human resources management solution including features like time tracking, e-learning, applicant management and payroll accounting.

[9] http://www.skills-base.com/
[10] http://www.infoniqa.com/hr-software/skill-management
[11] http://www.infoniqa.com/hr-software/personalmanagement

3.5.3. SkillsDB Pro

SkillsDB Pro[12] is an application designed to serve as a database in an organization, providing an overview of every person's personal skills and training only to themselves and their supervisor. The search function is capable of searching for multiple skills combined with different logical operators which enables users to enter very sophisticated queries. Not only can users provide information about their skills but supervisors can also do this with the limitation that no employee can see their supervisor's rating about themselves. Information about motivation, like in the other examined systems, cannot be captured. Furthermore, only privileged users like supervisors can search for persons. Taking into consideration that SinnerSchrader needs a tool to enable everyone to find someone with a specific skill set, this is a serious disadvantage. SkillsDB Pro also offers features SinnerSchrader does not intend to use including the automatic generation of project reports based on plan succession and demands for assessments.

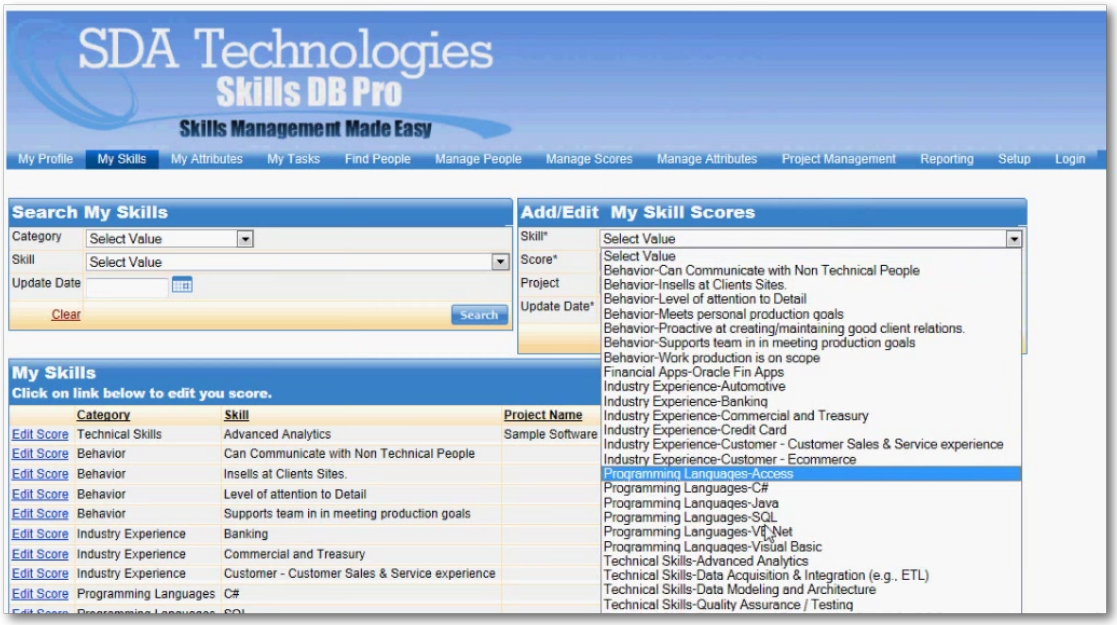

Figure 3.2.: SkillsDB Pro Overview (source: *http://skillsdbpro.com/*)

[12]http://www.skillsdbpro.com

3.5.4. Conclusion

None of the analyzed applications offer all required features, but all of them include various functions SinnerSchrader does not intend to use, which brings undesired complexity into the application. One of the most critical features, a search function that aggregates both knowledge and motivation is not offered by any of the commercial solutions. Furthermore, all those systems differentiate between employees and their supervisors and thus restrain transparency. The application is not supposed to be used for monitoring and rating employees, but should give employees the possibility to find each other; categorizing them into different roles would clearly defeat this purpose.

Pain Point Fitness Scoring

As shown by Canós-Darós, motivation is a vital factor regarding any employee's performance and quality of work [Can13]. Although motivation is a complex construct of many highly diverse dimensions, the size of the intersection of a person's interests and their duties is a key aspect to it. Assuming that every member of the company has some skills they prefer to employ over others, matching people to tasks that require the exact same abilities they are interested in employing will lead to more motivated employees and thus have a positive impact on the overall productivity. Consequently, when searching for persons having specific skills, the application should not only take into account the employees' skills but also their preferences in order to not find the most skilled, but the best fitting one. Unfortunately, none of the examined applications provide a way to aggregate both skills and preferences into a single score indicating the overall grade of suitability of a person relative to the searched skills.

4. Concept

The application should be accessible to all employees of SinnerSchrader. Due to the heterogeneity of the users' computer setups running *Windows*, *macOS* and *Linux*, creating a native application supported by everyone's system is a rather complicated task. A web application using standard technologies does not only solve this problem, but can also be used from mobile devices such as smartphones and tablets. Furthermore, there is no need to manually install and update the software on the client devices so that it can be assumed that all users use the latest version of the application. This is a positive factor regarding the overall usability of the system and assures bugs and security issues are eliminated the moment a new version of the software is deployed. All those advantages compared to native clients and the fact that SinnerSchrader's expertise lies in the development of web applications lead to the decision that such an application would be the appropriate choice.

4.1. Person Search

The central feature of the application is the search function that returns a list of all persons matching the entered set of skills. By default, the results are ordered by a *fitness score* which describes how well a person fits into the set of searched skills. As the system searches a set of data for a search query entered by the user to return matching items, it falls within the group of *information retrieval systems*.

4.1.1. Information Retrieval Systems

Information retrieval (IR) systems search a base of data (e.g. documents or websites) for attributes or keywords the user has entered. Primitive IR systems use search queries combined with boolean operators to specify the searched item, hence they are called *boolean systems*. The commercial solutions examined in 3.5 all use boolean systems. Modern IR systems "rank documents by their estimation of the usefulness of a document for a user query" [Sin01], which means they provide more suitable results without the need to enter complex, operator based search queries [Sin01].

4.1.2. Search Algorithm

The main feature of the application will be a person search function that uses a ranked IR system to retrieve the most suitable employee for a user's search query. In the context of this function, all employees are searchable items; their attributes include name, location and their respective skills structured as pairs of skill level and will level. The users will be able to enter the skills they search for and to specify an office location to search at. Minimum levels for skill and will cannot be specified because this would overcomplicate the search query. The IR system will find all employees that offer all skills entered by the user and then rank them so that the most suitable result will be presented first. Due to this ranking approach, the used IR system falls into the category of *Ranked Information Retrieval Systems*.

Outline

The basic structure of the IR system's working consists of five steps:

1. Create a list of all employees
2. Filter by Skills
   Remove all employees from the list that do not have all skills the user searched for. At this point, only the presence of the skill in the employees' profiles is taken into account; skill/will levels are ignored.
3. Filter by Location
   If the user specified a location, remove all employees from the list that do not match it.
4. Assign Fitness Scores
   Assign a fitness score to all remaining employees. This fitness score takes into account the user's skill/will levels and their specializations.
5. Sort by fitness score
   The results will be sorted by fitness score. The employee with the best fitness score will be shown first in the list of results.

Pseudo-Implementation

```
function search(searchItems, searchLocation) {
  var results = getAllEmployees()

  for (Employee e in results) {
    if (!e.skills.containsAll(searchItems)) {
      results.remove(e)
    }
  }

  for (Employee e in results) {
    if (e.location != searchLocation) {
      results.remove(e)
    }
  }

  for (Employee e in results) {
    e.assignFitnessScore()
  }

  results = results.sortByFitnessScore()

  return results
}
```

Figure 4.1.: Pseudo-Implementation of the search algorithm

4.1.3. Scoring Algorithm

The application will rank all found persons by their fitness into the searched skill set; this fitness will be scored on a scale from zero (worst) to one (best). The requirements and design of the algorithm calculating this fitness will be explained in this section.

Requirements

According to Spoonamore et al., an algorithm that matches persons to positions that are defined by the skills that are required to fill them has to meet more demands than solely the functional ones. They define the specific requirements such an algorithm assigning naval personnel to positions on a ship has to fulfill as follows:

> - Easy to implement and maintain
> - Fast to execute, so as not to become a computational bottleneck
> - Takes into account factors: rating, pay grade and NECs[1] and future taxonomies characterizing required knowledge, skills and abilities
>
> [SSH07, P. 14]

These qualities include factors very specific to the *US Navy* and thus will have to be evaluated and translated into SinnerSchraders' field of operation, but general requirements such an algorithm has to meet can be deduced: it may not be too complex as employees must be able to understand the system they are rated by, it should take into account different groups of factors and must be easy to adjust in order to keep the system maintainable.

Factors to Include

An estimation of a concrete person's fitness into the searched skill set needs to not only take into account the matching of offered and required skills, but also the employees' motivation to apply said skills, derived from their preferences and their personal specialization. The latter can be described as the difference between the person's skill/will levels in the searched skill set and their skills that are not part of the search query. In conclusion the factors that will be included in the algorithm are:

- Average level of knowledge regarding the searched skills
- Average level of will regarding the searched skills
- Specialization in the searched skills, derived from:
  - Specialization in knowledge about the searched skills
  - Preference of the searched skills over others

Proposed Fitness Score Algorithm

As shown in 3.2.3, the skill and will levels will be described on a four step scale, which will be expressed by natural numbers[2] between zero and three:

$$V = \{x \in \mathbb{N} \mid 0 \leq x \leq 3\}$$

All existing skills are accumulated in the set S. The employee's skills are represented by E which is a subset of S. The searched items are defined as Q.

$$S = \{Java, Ruby, C++, ...\}$$
$$E = \{x \in S \mid \text{employee has skill x}\}$$
$$Q = \{x \in S \mid \text{user searches for skill x}\}$$

The function v_s assigns a value of skill to any item in E; the function v_w assigns the respective level of will to any item in E.

$$v_s : E \mapsto V$$
$$v_w : E \mapsto V$$

[2]Includes zero as defined by ISO 80000-2

The scale values map to terms that describe the person's knowledge or interest:

Value	v_s	v_w
0	novice	uninterested
1	basic knowledge	indifferent
2	advanced knowledge	somewhat interested
3	expert	highly interested

The averages of the employees' skill/will values for the searched skills are defined as a_s and a_w. The variables s_s and s_w describe the employee's specialization in the searched items and are defined as the difference of the average skill/will level of the searched items and the average level of all other items. A person with maximal interest and knowledge in all searched items and the lowest ratings in their other items would have the greatest specialization possible and thus get assigned a value of one.

$$a_s = \left(\sum_{x \in E \cap Q} v_s(x) \right) \cdot \frac{1}{|E \cap Q|}$$
$$a_w = \left(\sum_{x \in E \cap Q} v_w(x) \right) \cdot \frac{1}{|E \cap Q|}$$

$$s_s = \frac{max(V) \ + a_s - \left(\left(\sum_{x \in E \setminus Q} v_s(x) \right) \cdot \frac{1}{|E \setminus Q|} \right)}{2max(V)}$$
$$s_w = \frac{max(V) \ + a_w - \left(\left(\sum_{x \in E \setminus Q} v_w(x) \right) \cdot \frac{1}{|E \setminus Q|} \right)}{2max(V)}$$

The resulting fitness score f is a weighted mean of the introduced factors. The weights $w_{as}, w_{aw}, w_{ss}, w_{sw}$ are positive real numbers and sum up to one.[3]

$$f = \frac{w_{as} \cdot a_s}{max(V)} + \frac{w_{aw} \cdot a_w}{max(V)} + w_{ss} \cdot s_s + w_{sw} \cdot s_w$$
$$w_{as} + w_{aw} + w_{ss} + w_{sw} = 1$$
$$w_{as}, w_{aw}, w_{ss}, w_{sw} \in \mathbb{R}^+ \cup \{0\}$$

[3]Mathematically, this is not necessary, but it results in much more human readable fitness score values between zero and one.

Example Calculation

Let there be three example employees, *Alice*, *Bob* and *Charlie*, having the same three skills each. (Notation: [skill level] | [will level])

Person	Java	Ruby	C++
Alice	2 | 1	2 | 2	3 | 3
Bob	2 | 3	0 | 3	0 | 1
Charlie	3 | 3	2 | 1	1 | 2

Applying the algorithm with $w_{as} = w_{aw} = w_{ss} = w_{sw} = 0.25$ to search for the skills *Java* and *Ruby* results in the following values[4]:

Person	a_s	a_w	s_s	s_w	f
Alice	2	1.5	0.33	0.25	0.44
Bob	1	3	0.67	0.83	0.71
Charlie	2.5	2	0.75	0.5	0.69

Ranking the employees only by the average value of skill regarding the two searched items would result in *Charlie* being preferred to *Alice* and *Alice* being preferred to *Bob*. Sorting them using the proposed fitness score, however, would result in *Bob* being recommended as the best match because his relatively high interest in the searched skills and his specialization in them compensates his low average skill. Interestingly, *Alice* has a better average skill level than *Bob* but nonetheless gets scored the worst due to her obvious specialization in C++. In real life usage, the weighting constants w_{as}, w_{aw}, w_{ss} and w_{sw} might need to be adjusted.[5]

[4] Values have been rounded off to two significant digits.

[5] The need to adjust the weights should not be considered a flaw since the algorithm has been intentionally designed to be customizable to the users' needs as defined in 4.1.3.

4.1.4. Example Search

For this example, let the set of employees be *Alice*, *Bob*, *Charlie*, *Donald*, and *Erika*, and the set of all known skills be *Java*, *Ruby*, and *C++*. The assignment of skill/will levels and the respective locations are:

Person	Location	Java	Ruby	C++
Alice	Hamburg	2|1	2|2	3|3
Bob	Hamburg	2|3	0|3	0|1
Charlie	Hamburg	3|3	2|1	1|2
Donald	Hamburg	3|3	-	2|2
Erika	Frankfurt	1|1	2|3	3|1

Let the weights used in the fitness score be $w_{as} = w_{aw} = w_{ss} = w_{sw} = 0.25$. Applying the algorithm and searching for employees knowing *Java* and *Ruby* in *Hamburg*:

- Create a list of all employees
  ⇒ Alice, Bob, Charlie, Donald, Erika
- Filter by skills: Donald gets eliminated because he does not have the skill *Ruby*
  ⇒ Alice, Bob, Charlie, Erika
- Filter by location: Erika works in Frankfurt and thus will be excluded
  ⇒ Alice, Bob, Charlie
- Assign fitness scores[6]
  ⇒ Alice (0.44), Bob (0.71), Charlie (0.69)
- Sort by fitness score
  ⇒ Bob (0.71), Charlie (0.69), Alice (0.44)

4.2. Recommender Systems

Recommender systems "are information filtering systems that deal with the problem of information overload by filtering vital information fragment *[sic]* out of large amount of [...] information" [IFO15] and are commonly used to recommend items to the user based on their previous interactions with other items. For example, recommender systems are used to predict products a customer might want to buy based on the ones they already bought in order to present those items more prominently than articles the customer is unlikely to fancy. In this application's context, two recommender systems will come to use in order to enrich the user experience by providing additional skills to search for (see 4.2.2) and by listing users similar to the currently shown one when the user inspects an employee's profile (see 4.2.3).

[6]The calculation of the fitness scores can be found in 4.1.3

4.2.1. Techniques of Content Filtering

As described by Isinkaye et al., filtering techniques used in recommender systems are divided in three classes: content based, collaborative, and hybrid. Each of these classes relies on a different approach for gaining data by which the information is filtered [IFO15].

Content Based Filtering

The content is filtered by examining its attributes in order to find items that are substantially similar to the one the user is currently or has previously been interacting with.

Collaborative Filtering

Collaborative filtering techniques rely on the assumption that users can be divided into groups of *neighbors* that behave similarly, so that recommendations are deductible from other users' former interactions.

- Model Based Filtering
  Model based filtering applies methods of machine learning and data mining to learn a precomputed model which predicts the users' interactions.
- Memory Based Filtering
  Memory based filtering techniques employ the saved interaction history and generate recommendations based on it. In contrast to model based filtering, memory based filtering does not learn a given model but operates directly on the saved data.
  - User Based Filtering
    The user's interactions with items are examined in order to find neighbors that share a similar activity history. Once neighbors are found, the system processes their interaction histories in order to find items the user is likely to appreciate getting recommended.
  - Item Based Filtering
    Item based filtering combines all users' interactions and creates a model describing which items are similar to another. This model is then used to recommend items similar to the ones the user has given positive feedback for.

Hybrid Filtering

Hybrid filtering combines two or more filtering methods either by aggregating their respective results into a single set of recommendations preferring the items multiple methods recommend, or by bringing content based aspects into the approach of collaborative filtering and vice versa.

4.2.2. Search Suggestions

After entering a skill to look for into the search field, the user will be presented other items they are likely to enter next. Since all searchable objects (all skills) are filtered in order to retrieve objects the user might want to interact with, which then will be presented to them proactively, this feature matches the definition of a recommender system given in 4.2.

Available Data

It is not planned that users will have to log in before performing a search, so there is no user context that can be used to examine a person's former interactions in order to predict and recommend their next one.
As the system is designed to be a web application, a cookie holding a unique identifier could be stored on the client device. The application would then use this ID to aggregate interactions made by the same person. Unfortunately, this method cannot identify a known person using another device because multiple devices will not share the same ID. Furthermore, data collected about a user will be discarded if they delete their devices' cookies or switch browsers. This approach would also need the application to inform the user that data will be stored on their devices as stated by Article 15(3) of the Telemediengesetz (TMG). The user has to give their approval and must be able to refuse the saving of their data (BDSG, Section 4a).
There also are various methods to identify users without the need to store any data on their devices by examining and recognizing their devices' attributes. The collected data can include factors like language settings, the used browser and its version, and the hardware components of the device. All this data combined can be used to form an almost unique fingerprint suitable to recognize a device [WGGK16].
Another possible method would be to recognize users by examining their very own behavior such as typing patterns or mouse strokes. This approach called *user fingerprinting* does not depend on the user's device and thus can be used to identify people across multiple devices and browsers. On the downside, this method can only differentiate between users typing the same word, it needs a multitude of samples of each user in order to be able to recognize them, and it is very failure-prone [AJL+05]. Although device and user fingerprinting are not prohibited by law, the *Opinion 9/2014 on the application of Directive 2002/58/EC to device fingerprinting* by the EU's *Article 29 data protection working party* states that a user must be informed about the fingerprinting process and be able to deny this. In the development of this recommender system, it will be assumed that there is no data about unique users, but about the aggregated entirety of the user base.

Skill Attributes

The skills are saved as simple names not enriched with any metainformation, so using content based filtering is not a trivial task. One possible approach would be to use linguistic methods to find similarities in names of skills in order to create clusters of related skills. Unfortunately, most of the skills' names are arbitrarily chosen or acronyms, so that this form of analysis will fail to detect any meaningful attributes.
Regarding the concept of the suggestion engine, the assumption is made, that there is no context to the skills and that the only information about any skill is its name.

Aggregated Search History

Tracking which skills have been searched together can be implemented easily and creates a fair amount of data to generate potentially useful suggestions. Legally, this is not problematic if the application does neither save personal data about the users (Article 15 Telemediengesetz) nor store information that could potentially be used to create personal usage profiles that can be matched to specific persons (Article 15(3) Telemediengesetz). Grouping skills that have been searched together does not stand in conflict with those regulations and requires no information about distinguishable user, so the application will store the search history for each item.

Chosen Approach

Assuming that no user context exists, user based filtering and model based filtering cannot be applied. Due to the lack of metadata about the skills, content based filtering can also be eliminated as a possible approach. Item based filtering, however, does not require any data that is unavailable, so this approach will be used for the recommender system.

Concept

The application has access to the list of skills the user has already entered into the search field and to a repository of all previous searches. Having this information, the system will use a markov chain to predict the next items that the user is likely to enter and recommend it to them. Markov chains are stochastic models predicting future states of systems based on the current state. In fact, markov chains rely on the fact that the next state of the system is exclusively dependent on the current state, which is called the *markov property* [Sie]. In the context of the skill management application, this is assumed to be true because only two states will be examined: the current state is represented by the set of all items entered in the search field; the future state is the skill set of the current search plus the item the user is going to enter next. The basic concept is to store all possible states of the system and the respective probability of switching from any state to any other one. Knowing the current state one can easily deduce the most probable future state. When a state transition occurs, the outgoing probabilities of the origin states can be adjusted accordingly in order to factor the transition into the prospective projections.

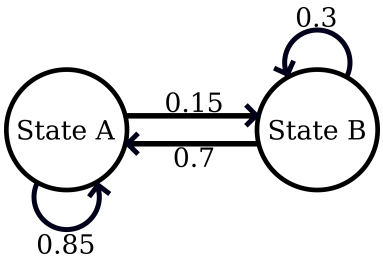

Figure 4.2.: A simple markov chain displayed as a graph. The states are represented as vertices. All possible transitions between states are denoted as edges. The edge weights define the probability of the transition relative to all other outgoing edges of a node.

Data Structure and Algorithm

An intuitive approach to the implementation of the markov chain generating the search suggestions would be realized by creating one state for every possible combination of skills to search for. State transitions describe the adding of a item to the set of already searched skills, so only transitions from a state to any state that describes the sames skill set plus one more search item are allowed. This approach would result in an exorbitant amount of states (for n skills, there would be $n!$ states) that are overly specific to a single search query, so that recommendations would be based on a small number of previous searches for the exact same combination of skills. As a tradeoff between keeping the number of states in the markov chain small and generating recommendations specific to the searched skill set, the system will generate predictions for each skill in the search query independently and aggregate these predictions afterwards. For each skill, a list of possible recommendations paired with the total count of searches for both will be stored. The recommendation lists of all skills combined represent the transition matrix. Instead of transition probabilities, the total number of searches is saved in order to simplify the aggreation of multiple suggestions.
The recommender system will concatenate the suggestion lists of all items in the current search query and add up the count numbers of skills appearing in multiple suggestion lists. Then, all elements of the combined list that are part of the search query will be removed, the result is a list of suggestions for the whole search query. The items will then be sorted by said search count. The first n elements, where n ist the number of items to recommend, will be returned.

Pseudo-Code

```
var knownSkills = [
  {
    name: "java",
    searchedWith: [
      {
        name: "php",
        count: 3
      }, {
        name: "ruby",
        count: 2
      }
    ]
  }, {
    name: "php",
    searchedWith: [
      {
        name: "java",
        count: 3
      }, {
        name: "ruby",
        count: 5
      }
    ]
    name: "ruby",
    searchedWith: [
      {
        name: "java",
        count: 2
      }, {
        name: "php",
        count: 5
      }
    ]
  }
]
```

Figure 4.3.: Pseudo-Implementation of the known skills' data structure

```
function suggest(searched, n) {
  var accumulated = {};

  for (s in searched) {
    for (t in knownSkills.getByName(t).searchedWith) {
      if (accumulated.getByName(t) exists) {
        accumulated.getByName(t).count += t.count
      } else {
        accumulated.getByName(t) = t.clone()
      }
    }
  }

  for (t in accumulated) {
    if (searched contains t) {
      remove t from accumulated
    }
  }

  return accumulated.sortByCount().getSubList(0, n)
}
```

Figure 4.4.: Pseudo-Implementation of the suggestion of a known skill based on a set of already searched items.

Example

Let there be the following transition matrix:

	Java	PHP	CSS	COBOL
Java	-	7	3	1
PHP	7	-	9	5
CSS	3	9	-	8
COBOL	1	5	8	-

In this example, the query "Java, PHP" has already been entered. Based on those elements, one more item shall be recommended to the user ($n = 1$).

1. Retrieve suggestion lists for each search item
   ⇒ PHP (7), CSS (3), COBOL (1), Java (7), CSS (9), COBOL (5)

2. Aggregate lists (combine counts)
   ⇒ PHP (7), Java (7), CSS (12), COBOL (6)

3. Remove suggestions that are part of the search query
   ⇒ CSS (12), COBOL (6)

4. Sort by count
   ⇒ CSS (12), COBOL (6)

5. Suggest the first n items ($n = 1$)
   ⇒ CSS

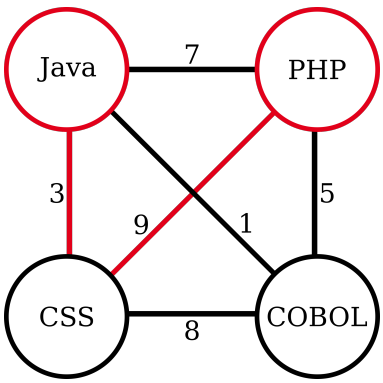

Figure 4.5.: Markov Model used in the example. The two origin states and the transitions that form the end result are highlighted red.

4.2.3. Recommending Similar Users

Additionally to the required features, the backend will provide the capability to find users that are similar to a reference person. In the application, this feature will be used to show recommended persons when viewing one specific employee's profile. As this functionality filters SinnerSchrader's entire workforce to find the ones that are similar to currently shown profile and then proactively recommends those people to the user, it is another recommender system. Unlike the recommender system used to provide more skills to search for, this system will use a content-based filtering approach. The employee profiles contain numerous skills that represent a sufficient amount of inherent information about the respective person to compare profiles in order to find similar ones. Other users' interactions with the profiles, however, do not necessarily indicate similarities, so that the interaction history will not be taken into account for this feature. As a consequence, collaborative filtering approaches are not applicable to this problem. The recommender algorithm will contain three straightforward steps:

- Get a list of all users except the reference one
- Sort the list by similarity with the reference user
- Recommend the first n items in the list (n = number of recommendations to make)

Jaccard Similarity Coefficient

As shown in 4.1.3, users can be described as sets of skills they have, so the task of measuring the similarity between users can be abstracted to estimating the correlation between their skill sets. A popular approach to this problem is using the *Jaccard Similarity Coefficient*, a statistical metric to determine the similitude of two finite sets by dividing the size of their intersection with the size of their union [DS15]. The resulting value describes how similar the skill sets are and thus represents how similar the respective users are. The results' maximum value is one (users are identical); the minimum value is zero (users to not share any skills).
Let S be the set of all skills existing in the system. Persons will be described as the sets of skills they have: the employee based on whom the recommendations shall be made will be called R (*reference employee*). The employee whose similarity with R is to be measured will be called E.

$$S = \{Java, Ruby, C++, ...\}$$
$$E = \{x \in S \mid \text{examined employee has skill x}\}$$
$$R = \{x \in S \mid \text{reference employee has skill x}\}$$

The Jaccard Similarity Coefficient j is defined as the size of the intersection of E and R divided by their union:

$$j(E) = \frac{|E \cap R|}{|E \cup R|}$$

Example Calculation

In this example, *Alice* will be the reference employee. Four other employees, *Bob*, *Charlie*, *Donald*, and *Erika*, will be the set of persons to pick recommendations from. The set of existing skills will be *Java*, *Ruby*, *PHP*, *.NET*, and *CSS*. The goal is to recommend two persons to a user who is inspecting *Alice's* profile. The employees' skill sets are[7]:

Employee	Java	Ruby	PHP	.NET	CSS
Alice	✓	✓		✓	
Bob	✓			✓	✓
Charlie		✓	✓	✓	
Donald	✓		✓		
Erika	✓	✓	✓		✓

The definitions can be applied; since the goal is to find two people similar to *Alice*, her skills are represented by R.

$$S = \{Java, Ruby, PHP, .NET, CSS\}$$
$$R = \{x \in S \mid \text{Alice has skill x}\} = \{Java, Ruby, .NET\}$$

For each employee, the Jaccard Similarity Coefficient will be calculated based on the union and intersection of their respective skill set and *Alice's* skills.

Employee (E)	$E \cap R$	$E \cup R$	j_E
Bob	$\{Java, .NET\}$	$\{Java, Ruby, .NET, CSS\}$	0.50
Charlie	$\{Ruby, .NET\}$	$\{Java, Ruby, PHP, .NET\}$	0.50
Donald	$\{Java\}$	$\{Java, Ruby, PHP, .NET\}$	0.25
Erika	$\{Java, Ruby\}$	$\{Java, Ruby, PHP, .NET, CSS\}$	0.40

Given the task to recommend two persons that are similar to *Alice*, the recommender system would choose *Bob* and *Charlie* because they have the highest Jaccard Similarity Coefficient in the set of given employees.

[7] Notation: ✓ ⇒ employee has skill

Drawbacks

The described recommender system inspects how many skills the examined persons have in common relative to their total number of skills. The levels of knowledge and motivation regarding those skills are not taken into account, so the recommendations might be inaccurate. Unfortunately, there is no real life user feedback yet, so the evaluation of this factor and the creation of a implementation that includes those aspects will have to stay subject to further research.

4.3. Visual Concept

The application should be as simple as possible and usable for everyone in order to provide an efficient and fast tool. Thus, it will be designed as a single page application based around a search function that provides a way to input the skills to search for and returns all persons offering said skills. After entering a search, the user can select any of the found colleagues and view their personal profile showing extended information like contact details, more skills the user did not search for, and the employee's location. This profile will also include links to directly contact the inspected person via e-mail or *Google Hangouts*[8]. More information about the concept behind the visual design and the frontend's implementation is provided by Strecker [Str17].

[8]https://hangouts.google.com

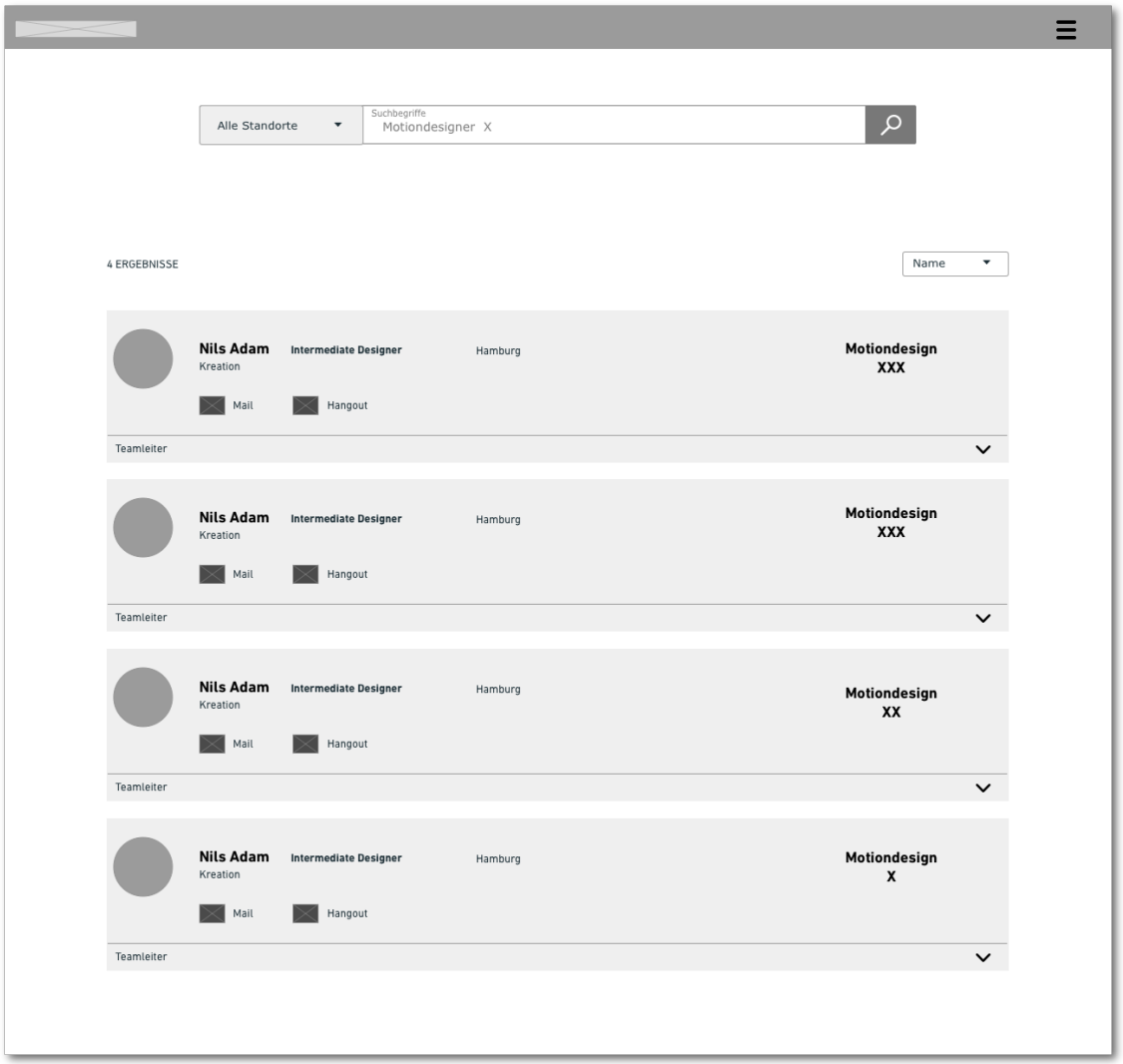

Figure 4.6.: Wireframe of the search result view

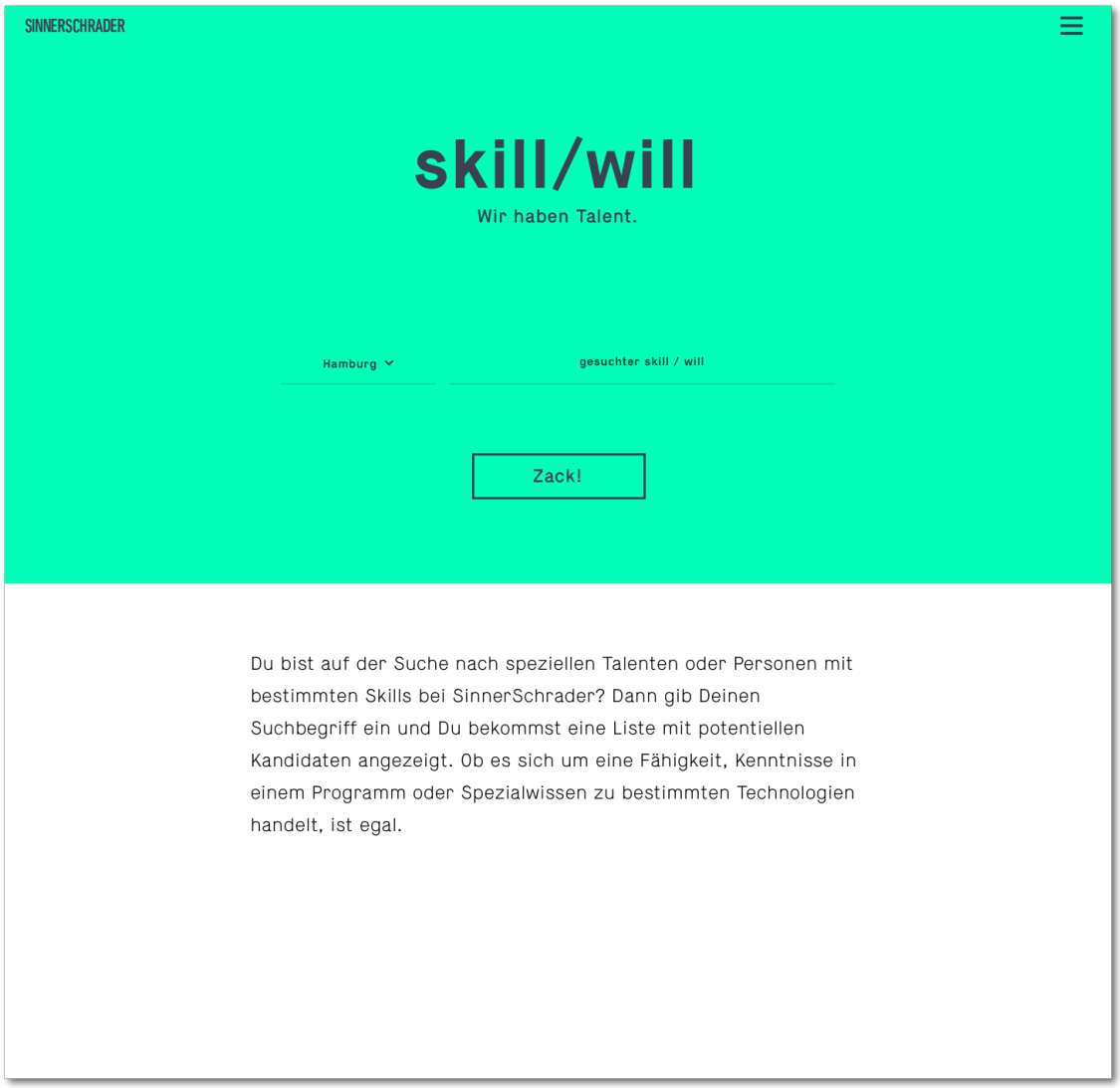

Figure 4.7.: An early prototypical design for the search view

4.4. Legal Concerns

The information saved in the system fall into the category of personal data (*Personenbezogene Daten*), which is defined as "any information concerning the personal or material circumstances of an identified or identifiable individual (the data subject)" (BDSG, 3(1)). The personal data will be collected (BDSG, 3(3)), processed (BDSG, 3(4)) and transferred (BDSG, 3(3)) to other employees of SinnerSchrader. Generally, this does not violate any law, since the "collection, storage, modification or transfer of personal data or their use as a means of fulfilling one's own business purposes shall be admissible" (BDSG, 28(1)) but some restrictions apply: the data subjects have to be informed about the processing of their personal data, they must be able to deny their consent (BDSG, 4a), and the personal data shall not be made public. To ensure the latter, the application must not be accessible to persons that do not work for or on behalf of SinnerSchrader. Technically, this will be arranged by making the application reachable from SinnerSchrader's internal network only which can exclusively be accessed by employees and authorized persons.
Furthermore, the Works Constitution Act (*Betriebsverfassungsgesetz*) defines the rights and roles of the works council (*Betriebsrat*) which have an impact on the design of the application. It states that "the works council shall have a right of co-determination in the introduction and use of technical devices designed to monitor the behaviour or performance of the employees" (BetrVG, 87(6)); since the application describes the employees' knowledge which is a key factor to their performance, this definition applies to it, so that the works council had to be involved in the design process. Regarding the possibility for users to rate each other's skills as described in 3.2.3, the council exercised its rights to object and interdicts its implementation because of both legal reasons and the personal concerns of employees; the council's statement can be found in appendix A.

5. Implementation

5.1. Application Structure

The application consists of two main components: the frontend that presents the user with a graphical interface, and the backend that provides data and actions on them to the frontend. The user's browser connects to a web server that acts as a reverse proxy and not only provides static resources like HTML and CSS files which represent the frontend, but also acts as an SSL endpoint. Requests for dynamic data and actions that are handled via the REST API provided by the backend are passed on to it, its response will then be directed through the reverse proxy to the client. To store and read data, the backend connects to a MongoDB database. User details are synced from the existing LDAP server which acts a central repository for personal information of all employees.

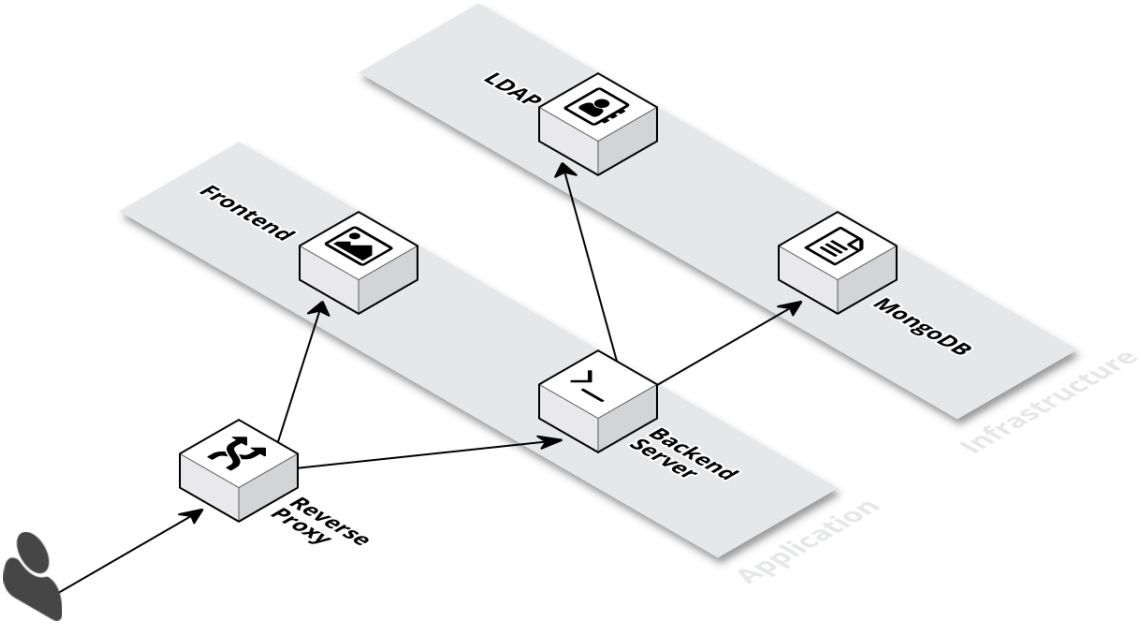

Figure 5.1.: The system's architecture. Created with *https://cloudcraft.co*.

5.2. MongoDB

MongoDB[1] is a popular non-relational NoSQL database that aims to be fast and easy to use [HMPH15, p. 10]. To increase performance, like many NoSQL databases, it does not provide ACID[2] transactions which are a well-known feature of relational database management systems (RDBMS). This, however, simplifies horizontal scaling since new machines can easily be inserted into an existing cluster of database servers without the need to be in sync [HMPH15, p. 3]. To compensate the lack of atomic operations, optimistic locking can be used to prevent concurrent writing operations [PRG+17].

5.2.1. BSON

In contrast to relational databases that store all data in tables, MongoDB uses a document-oriented data structure saving every element in the Binary JSON[3] (BSON) format. This approach allows complex data to be stored as one object rather than having to dissect its nested elements and storing them in separate tables. As a consequence, retrieving an object from the database is much more efficient than it would be using an RDBMS, as the latter needs to join the tables storing the object's nested sub-objects and compose the requested element whereas MongoDB stores it in the exact same form it is requested [HMPH15, p. 10].

5.2.2. Data Structure

The application stores three different object classes in the database: *skills* that are known to the system, *persons* with their individual contact data and personal skills, and *sessions* used to authenticate users that wish to modify their profiles. In order to instantiate the elements as Java objects, Spring Data[4], the framework used for database access, also stores the class which the object needs to be mapped to. Furthermore, every item has the field *version* which is created and managed by Spring Data and holds a version number used to resolve writing conflicts that may occur when multiple threads access the same object simultaneously.

[1] https://www.mongodb.com
[2] Atomicity, Consistency, Isolation, Durability
[3] Javascript Object Notation
[4] http://projects.spring.io/spring-data/

Known Skills

Skills known to the system consist of a unique name, a descriptor that will be used to generate an icon for each skill, and a list of suggestions that themselves are expressed by a name, the total count of searches of the respective suggestion together with the skill. This list will be used to predict the next item a user is likely to enter as described in 4.2.2.

```
{
    "_id" : "Java",
    "_class" : "[...].skills.KnownSkill",
    "iconDescriptor": "some icon",
    "suggestions" : [
        {
            "name" : "AEM",
            "count" : 1
        }, {
            "name" : "jquery",
            "count" : 1
        }
    ],
    "version" : NumberLong(3)
}
```

Figure 5.2.: Data structure of a skill stored in the database

Sessions

Sessions are used to authenticate users that wish to modify their personal profile. The client has to authenticate the user with their credentials; if this is successful, a new session holding a unique ID, the point of time on which it will expire, and the ID of the authenticated user, will be created and stored in the database.

```
{
    "_id" : "87163f310f124830bac677fe31484262",
    "_class" : "[...].session.Session",
    "username" : "foobar",
    "expireDate" : ISODate("2017-01-09T08:36:40.128Z"),
    "version" : NumberLong(1)
}
```

Figure 5.3.: Data structure of a session stored in the DB

Persons

The documents that represent persons contain the respective person's ID[5], their personal data like first and last name, telephone number, e-mail address, office location, job title[6] (e.g. "Senior Java Developer"), a comment section, and a list of the person's skills. Each of those skills consists of a name, a level of skill, and a level of will.

```
{
    "_id" : "foobar",
    "_class" : "[...].domain.person.Person",
    "skills" : [
        {
            "_id" : ".NET",
            "skillLevel" : 1,
            "willLevel" : 2
        },{
            "_id" : "Scrum",
            "skillLevel" : 3,
            "willLevel" : 1
        }
    ],
    "version" : NumberLong(1),
    "ldapDetails" : {
        "firstName" : "Fooberius",
        "lastName" : "Bartels",
        "mail" : "foobar@mail.org",
        "phone" : "+49 12 345678 901",
        "location" : "Hamburg",
        "title" : "Development"
    },
    "comment" : "Certified Cyber Specialist",
}
```

Figure 5.4.: Data structure of a person stored in the DB

[5]Each employee gets assigned an internal ID (*Benutzerkürzel*) that is globally used to uniquely identify a person.

[6]The job title data is not maintained consistently in the LDAP, so that, unfortunately, it is not suitable to be used in the person search.

5.2.3. Queries

As shown in 5.2.1, the document based data structure of MongoDB allows the database to efficiently perform complex requests. Furthermore, it provides simple and straightforward search queries to retrieve objects based on their attributes. For example, getting all users who offer the skill *Ruby* from the collection *person* can be done with this straightforward query:

```
1 db.person.find({ "skills._id" : "Ruby" })
```

Figure 5.5.: MongoDB query to retrieve all users with the skill *Ruby*

5.3. LDAP

SinnerSchrader runs an LDAP service which acts as a centralized source of personal information of all employees to provide all internal tools with data. It manages the employee's user accounts that are used in different internal services such as *Jira*[7], *Confluence*[8] and *Move*[9]. The application connects to it in order to retrieve contact information to display in users' profiles. In comparison with having the users to enter their data manually, this method has the benefit that the users' data will be kept in sync across all internal services, and that it reduces the effort a user has to spend to create their profile.

5.4. Reverse Proxy

Between the client and the backend, an intermediary web server that acts as a reverse proxy is switched in. Its main purpose is the distinguishing between requests for static files, like HTML and CSS content that will be directly delivered by said server, and API calls that are redirected to the backend. This increases the system's security by protecting the backend server's identity and presenting an additional defense layer [NGI]. Furthermore, this server can handle SSL encryption between the application and the client, and, if multiple backend servers are needed, balance the workload between them while presenting one uniform service to the outside (see 6.2.1).

[7] https://www.atlassian.com/software/jira
[8] https://www.atlassian.com/software/confluence
[9] An internally developed room management application.

5.5. API

To exchange data between the backend and the frontend, a *Representational State Transfer* (REST) API is provided by the backend. Its endpoints are called by the frontend code to either request data or to command the backend to perform modifying operations on it. The used HTTP method is the main indicator of the action to perform: *GET* is used to retrieve data, *POST* to insert new elements, *PUT* to modify existing ones and *DELETE* to remove them. The URLs of the individual action express the entity on which the action will be performed. All API endpoints are listed in table 5.1.

5.5.1. API Response Format

The API returns data in the JSON format, which is one of the two de-facto standards for data exchange on the web[10] because it is part of the Javascript (JS) language [Smi15, p. 37]. Approximately 94% of all websites use JS [QS]; since JSON directly represents JS objects and is both easy to parse and human-readable, it became the leading data format for web applications. For every HTTP request, the backend will return a JSON response, even though the request did not demand data to be returned. In this case, the response will contain status information about the success of the requested action.

```
{
    "id" : "foobar",
    "firstName" : "Fooberius",
    "lastName" : "Bartels",
    "mail" : "foobar@mail.org",
    "phone" : "+49 12 345678 901",
    "location" : "Hamburg",
    "title" : "Development",
    "comment" : "Certified Cyber Specialist",
    "skills" : [
        {
            "name" : ".NET",
            "skillLevel" : 1,
            "willLevel" : 2
        }, {
            "name" : "Scrum",
            "skillLevel" : 3,
            "willLevel" : 1
        }
    ]
}
```

Figure 5.6.: Example JSON response by the API for the request */users/foobar*. For comparison, the corresponding database entry is shown in 5.2.2.

[10] The other one is *XML* used by the *Simple Object Access Protocol*

URL	HTTP Method	Non-URL Parameters	Return Statuses	Comment
/login	POST	username, password	200, 401, 500	Try to login a user; returns session key
/logout	POST	session	200, 401, 500	Logout a session
/skills	GET	search	200, 500	Search for autocompletion; returns all skills if search is empty
/skills	POST	name	200, 400, 401, 500	Add new skill with the given name
/skills/next	GET	search, count	200, 400, 500	Suggest skills based on the search query
/skills/{skill}	DELETE		200, 400, 401, 404, 500	Remove the skill with the given name
/skills/{skill}	PUT	name	200, 400, 401, 404, 500	Rename the skill
/users	GET	skills, location	200, 400, 500	Get all users matching the search
/users/{user}	GET		200, 404, 500	Return the specified user
/users/{user}/similar	GET		200, 400, 404, 500	Get similar users
/users/{user}/details	PUT		200, 400, 404, 500	Modify the user's details
/users/{user}/skills	POST	session, skill, skill_level, will_level	200, 400, 401, 404, 500	Create new skill/modifiy existing personal skill
/users/{user}/skills	DELETE	session, skill	200, 400, 401, 404, 500	Remove the skill from the users profile

Table 5.1.: All API endpoints provided by the backend

5.6. Backend Implementation

The backend component is implemented in Java 8[11] using the Spring Boot framework[12]. Maven[13] is employed to manage the build process and run unit and integration tests.

5.6.1. Architecture

The software architecture consists of three main categories of classes: services that handle data manipulation and filtering and hold the business logic, repository objects that wrap the database operations into easy to use handlers, and domain specific data types. Additionally, numerous helper classes like custom exception types, comparators, and general utilities are implemented.

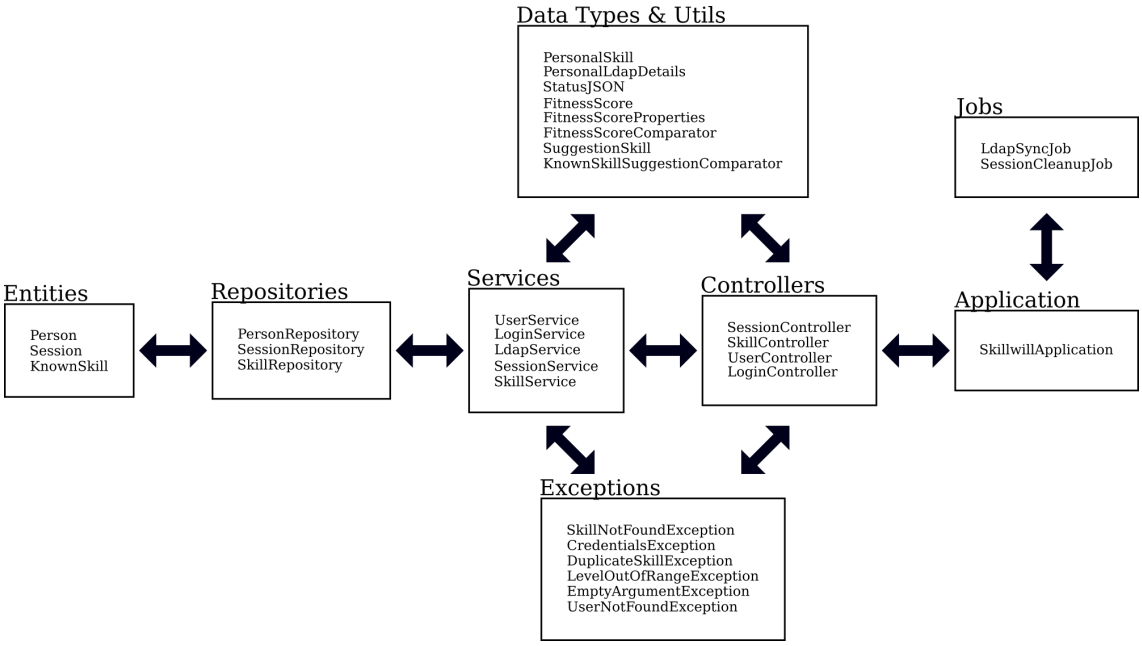

Figure 5.7.: Class structure of the application's backend software.

[11] https://go.java/
[12] https://projects.spring.io/spring-boot/
[13] https://maven.apache.org/what-is-maven.html

5.6.2. Spring Boot

Spring Boot is a sophisticated web framework that provides numerous features to create web applications including, but not limited to, annotations to expose Java methods as HTTP request endpoints, an embedded webserver[14] to run the application on, a modular design to extend its features, and dependency injection. It is being used to use because its credo to provide default configurations where possible and thus reduce the need to write infrastructure code simplifies the applications' structure [Gut16, p. 6]. For example, a controller that returns a static response can be created using two annotations: *@Controller* to make Spring Boot identify the class as a resource that will listen to HTTP calls, and *@Request* to specify the URL and HTTP method to use. Unlike most other web frameworks, Spring Boot does not require any more configuration or dispatching classes.

```
1  @Controller
2  public class HTCPCPImpl {
3
4      @RequestMapping(path = "/coffee", method = RequestMethod.GET)
5      public ResponseEntity<String> coffee() {
6          StatusJSON json = new StatusJSON("I'm a teapot \u2615");
7          return new ResponseEntity<String>(
8              json.toString(),
9              HttpStatus.I_AM_A_TEAPOT
10         );
11     }
12
13 }
```

Figure 5.8.: Example controller using Spring Boot

[14] Apache Tomcat 7 (http://tomcat.apache.org/)

5.6.3. Spring Data Repositories

Spring Data[15] is a module for Spring Boot that streamlines the way objects can be stored and retrieved from a database. The components used in this application are CRUD[16] repository objects that enclose the database connections and serve simple Java methods as an interface. To create such a repository, a Java interface defining the stored data type and custom database queries has to be constructed. No actual implementation of the interface has to be realized since it will be generated automatically by Spring Data. The parameters needed to connect to the database have to be configured in any source of properties known to Spring Boot, e.g. in *src/main/resources/application.properties*.

```
1  public interface PersonRepository
2          extends MongoRepository<Person, String> {
3
4      Person findById(String id);
5
6      @Query("{ 'skills._id' : '?0' }")
7      List<Person> findBySkill(String skillName);
8
9      @Query("{ 'skills._id' : { $all : ?0 } }")
10     List<Person> findBySkills(List<String> skillNames);
11
12 }
```

Figure 5.9.: Example for a repository interface managing person objects stored in the database.

```
1 spring.data.mongodb.host=127.0.0.1
2 spring.data.mongodb.port=27017
3 spring.datasource.driverClassName=com.mongodb.Mongo
```

Figure 5.10.: All configuration parameters needed to run Spring Data

5.6.4. LDAP Connection

To connect to the LDAP server, the *unboundid* library[17] which provides methods to open a TCP connection to the server, make requests, and parse the server's response comes to use. The connection to the LDAP server will be kept alive and is reused for all operations, so that the effort to open a new connection is minimized and to avoid networking problems with too many parallel connections.

[15] http://projects.spring.io/spring-data/
[16] Create, Read, Update, Delete
[17] https://www.ldap.com/unboundid-ldap-sdk-for-java

```java
@Service
@Scope("singleton")
@EnableRetry
public class LdapService {

    private static Logger logger =
        LoggerFactory.getLogger(LdapService.class);

    // [fields not used in this example]

    private static LDAPConnection ldapConnection;

    @Autowired
    private PersonRepository personRepo;

    // [methods for user sync and connection handling]

    public boolean canAuthenticate(String username,
            String password) {
        try {
            BindRequest bindRequest = new SimpleBindRequest(
                "uid=" + username + "," + ldapBaseDN, password);
            BindResult bindResult =
                ldapConnection.bind(bindRequest);
            if (bindResult.getResultCode()
                    .equals(ResultCode.SUCCESS)) {
                return true;
            }
            return false;
        } catch (LDAPBindException e) {
            return false;
        } catch (LDAPException e) {
            logger.error("Failed to authenticate: LDAP error");
        }
        return false;
    }

}
```

Figure 5.11.: LDAP user authentication using the unboundid library. Note: parts of the code have been removed to simplify this example.

5.6.5. Swagger

Swagger[18] is an open source framework for creating documentations of REST APIs. Its annotation based Java integration is used to generate an interactive overview of the API endpoints provided by the backend. This overview is automatically served by Spring Boot and contains a list of all URLs to make requests to, HTTP response codes to expect, the content type of the response, and a built-in form to make example requests. The main advantage of this approach is that the code and its documentation are located at the very same place and that parts of the documentation are generated automatically, so that both are maintained synchronously, thus avoiding the documentation differing from the implementation.

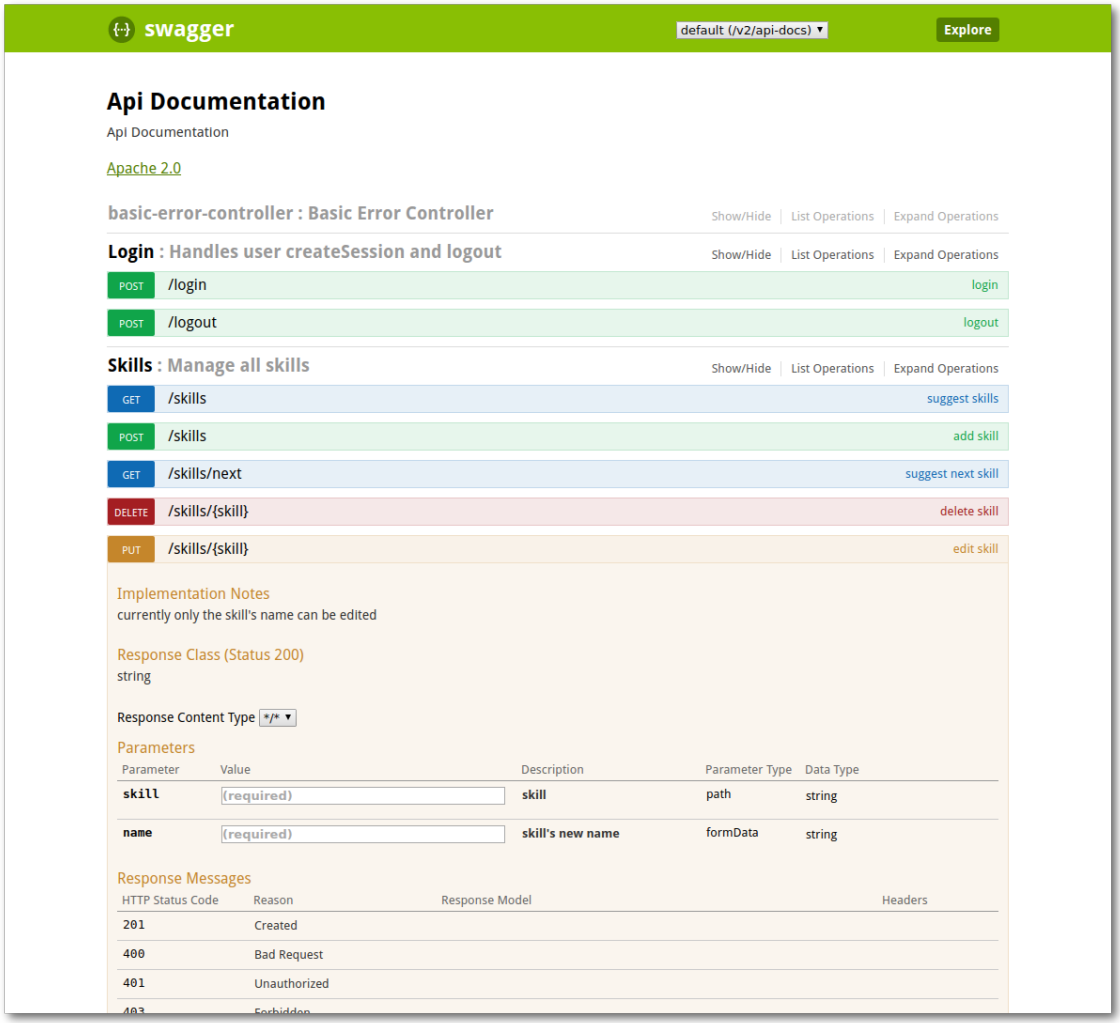

Figure 5.12.: Interactive API documentation generated by Swagger

[18] http://swagger.io/

5.6.6. Testing

As a part of the build process, automatic tests are run using the *JUnit*[19] framework. Two types of tests are employed to ensure the proper working of the software: unit tests that validate isolated segments (Java classes), and integration tests that simulate calls to the controllers and test the interplay of the individual components.

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest
public class UserControllerTest {

    @Test
    public void testGetUsersValid() throws JSONException {
        logger.debug("Testing UserController: get valid users");
        ResponseEntity<String> res =
            userController.getUsers("Java", "Hamburg");
        assertTrue(res.getStatusCode() == HttpStatus.OK);
        assertTrue(new JSONArray(res.getBody()).length() == 1);
        assertTrue(new JSONArray(res.getBody())
            .getJSONObject(0).has("id"));
        assertEquals("foobar", new JSONArray(res.getBody())
            .getJSONObject(0).getString("id"));
    }

}
```

Figure 5.13.: Example unit test using JUnit (Multiple methods removed for this example)

Embedded Services

During the integration test phase, external services like LDAP and a database have to be accessed in order to ensure the proper working of the interfaces connecting to them. Using the live services, however, is not an option as it cannot be assumed that the machine that runs the tests has a connection to them, and because the tests have to take control over the state of the services. To solve this, an LDAP server and a MongoDB are embedded into the application and will be used during testing. The embedded database is the MongoDB implementation by *flapdoodle*[20], which has the advantage of being effortlessly deployed by importing it; all further configuration and setup happen automatically. To embed an LDAP service, the *unboundid* library is used.

[19] http://junit.org
[20] https://github.com/flapdoodle-oss/de.flapdoodle.embed.mongo

5.7. License

The software is licensed under the MIT license [Mas88] which is considered one of the most popular open source licenses [Bal], mainly because it grants a high level of freedom to modify and use the software under the sole condition that a copy of the original license is distributed algonside the software.

> Copyright 2017 SinnerSchrader Deutschland GmbH
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

[Mas88]

6. Evaluation

6.1. Fitness Score Algorithm

6.1.1. Uniform Distribution of Score Values

The values calculated by the fitness score algorithm (see 4.1.3) should be distributed uniformly in the complete range of possible values because this does not only generate the best search results, but also is an indicator for the algorithm's fairness. To test this, one hundred automatically generated persons have been fed into the system. Each test person has been assigned a random number of skills between zero and 17, with random skill and will levels each, thus a search for any skill will return a list of up to one hundred persons sorted by their fitness score. The search results for the skills *Atomic Design*, *Datenbanken*, *Funkspots*, *hybris*, *Kommunikation*, *MySQL*, *Sketch* and *Text* have been analyzed because those have the highest number of results (33 each); all results can be found in appendix C. Ideally, the scores in each result list decline linearly from one to zero. The ideal value for the n-th result[1] value is:

$$f_{Ideal}(n) = 1 - \frac{n-1}{32}$$

The distribution of the score values for every respective search is shown in figure 6.1. Figure 6.2 illustrates the average of all eight scores for each position in the search result lists, the maximum value found at this position, and the respective minimum value. It shows that the average score declines approximately linearly, which means that the score values are distributed uniformly thorough the result lists. In figure 6.1, every result function shows a variance from the ideal value; the average value in figure 6.2, however, deviates significantly less from the ideal than any of the isolated data rows. This leads to the conclusion that the drift from the ideal that occurs when observing one single set of data for one specific search is based on the small amount of examined values (33 data points) and the individual deviations compensate each other. The average fitness score shows an mean error of approx. 6% (see figure 6.3). The maximum error in the given set of data is 27%. This leads to the conclusion that the fitness score algorithm generates uniformly distributed score values with an acceptable error.

[1] Counting starts at one

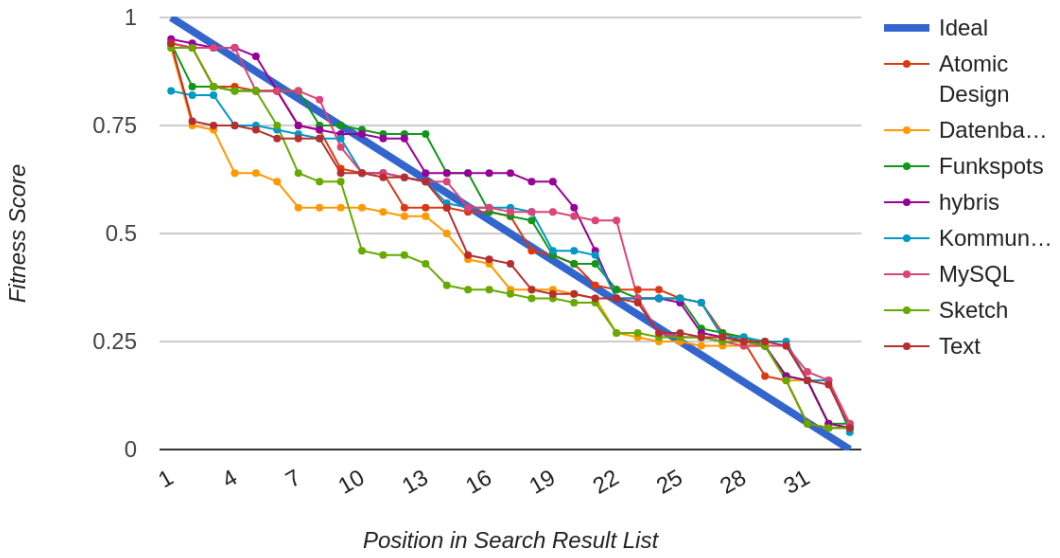

Figure 6.1.: All search results and the ideal value.

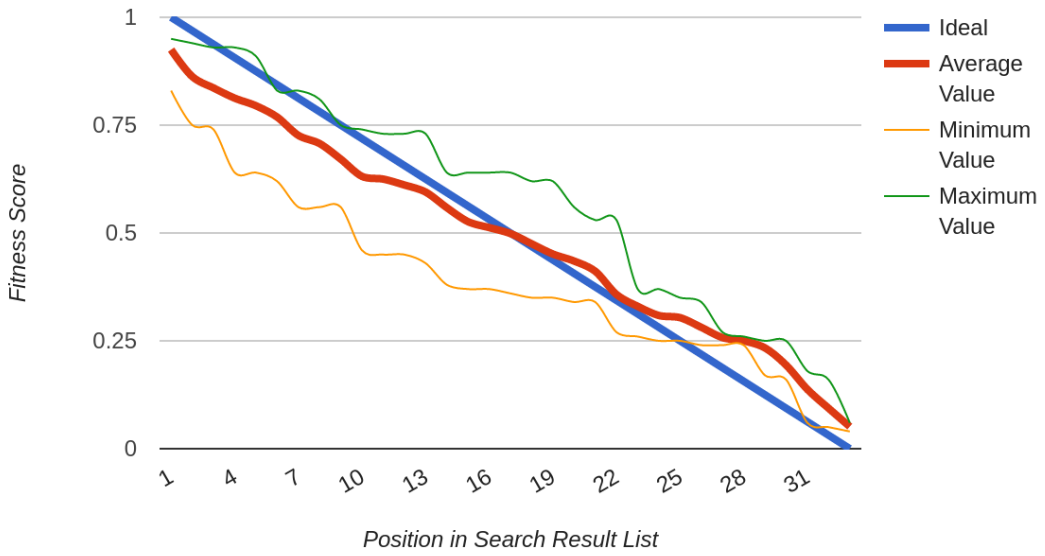

Figure 6.2.: Average, maximum and minimum score for each position in the respective search result list.

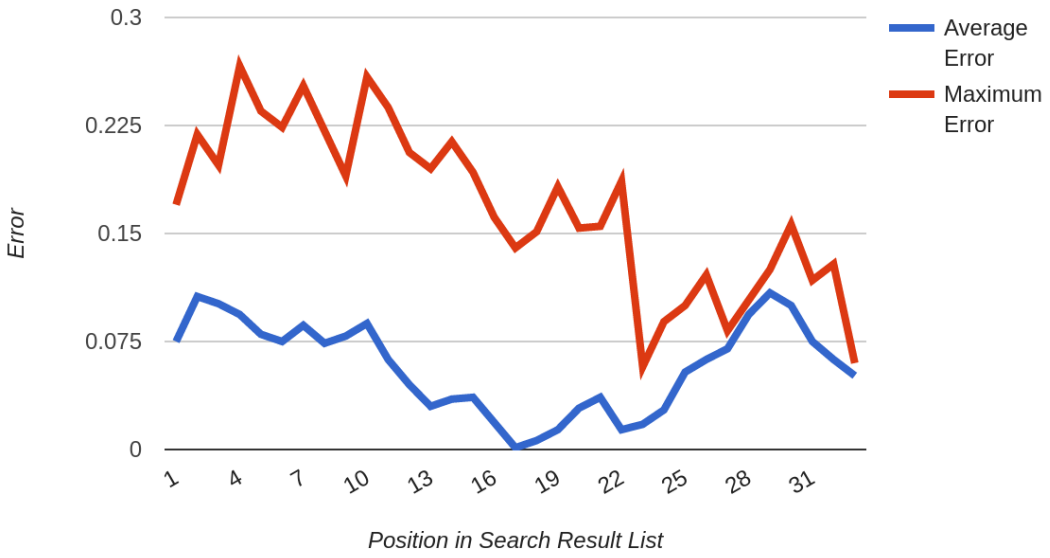

Figure 6.3.: Average and maximum error (Difference between respective score value and the ideal value).

6.1.2. Fitness Score Algorithm vs. Human Estimations

The algorithm is supposed to calculate a person's fitness so that its results match the ratings estimated by other employees. To validate that the chosen approach is capable of this, a set of fictional employees has been rated by both the algorithm and employees. The respective estimations have been compared to analyze if there is a configuration of weigthing parameters w_{as}, w_{aw}, w_{ss} and w_{sw} that makes the algorithm produce scores that are congruent to the estimations made by the rating persons.

Examined Test Records

For this test, five test records have been examined: *Alice*, *Bob*, *Charlie*, *Donald* and *Erika*. Each of these fictional persons has different skill and will levels for the abilities *Java*, *AEM*, *Ruby* and *.NET*. The fitness scores that have been collected and examined represent the scenario that a potential user searches for the skills *Java* and *AEM*.

Test Record	Java	AEM	Ruby	.NET
Alice	3 | 3	2 | 3	0 | 1	2 | 2
Bob	2 | 1	3 | 0	2 | 0	3 | 3
Charlie	1 | 3	0 | 2	1 | 2	2 | 3
Donald	1 | 2	2 | 1	1 | 2	2 | 1
Erika	1 | 0	0 | 1	3 | 2	3 | 1

Table 6.1.: Skill and will levels of the persons presented in the survey. Notation: [skill level] | [will level]

Survey

A random group of 161 employees (35% of SinnerSchrader's staff) have been presented a survey using *Google Forms*[2]. In total, 41 persons have given their responses in a timeframe of 72 hours. The survey consisted of two sections: the estimation of the test persons' fitness scores and the evaluation of the weighting of the factors included in the algorithm. To collect the personal estimations regarding the test records' fitness, the test subjects have been presented *Likert-Type Items*[3] using a scale from one ("does not fit at all") to five ("perfect match"). As table 6.2 shows, values on this scale can easily be translated into the corresponding fitness score. The results of the survey are shown in table 6.3[4] and figure 6.4.

[2] https://forms.google.com
[3] *Likert-Type* as defined by Uebersax [Ueb].
[4] Values have been rounded off to two significant digits.

Survey Rating	1	2	3	4	5
Fitness Score	0	0.25	0.5	0.75	1

Table 6.2.: Conversion from survey rating to fitness score.

Test Record	1	2	3	4	5	-	Mean	f
Alice	0	0	0	15	26	0	4.63	0.91
Bob	5	26	9	1	0	0	2.15	0.29
Charlie	1	14	23	3	0	0	2.68	0.42
Donald	0	10	27	3	0	1	2.83	0.44
Erika	32	70	0	2	0	0	1.31	0.08

Table 6.3.: Fitness scores estimated by 41 test subjects. (Total count per rating scale item)

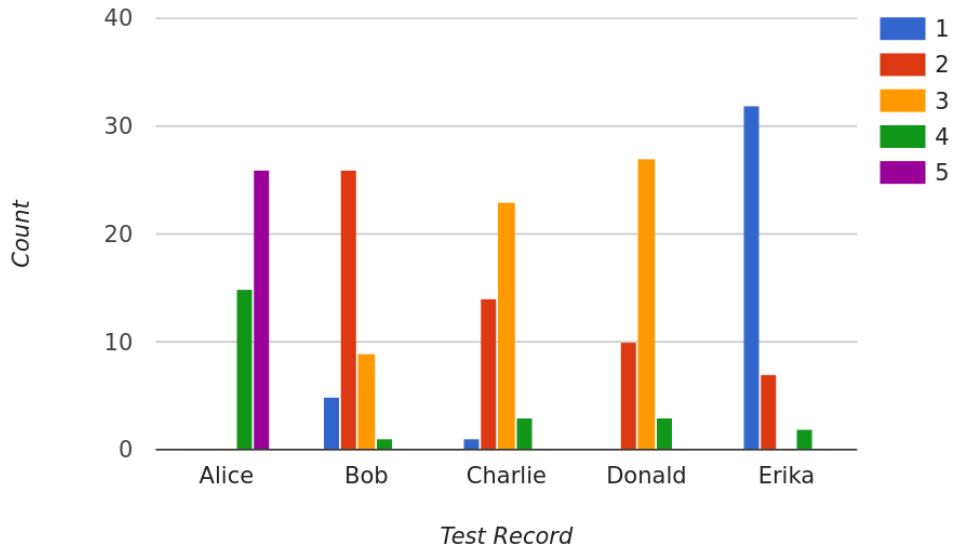

Figure 6.4.: Data collected in the survey.

Furthermore, the participants were asked to rate the importance of the four factors included in the algorithm (see 4.1.3), namely the person's average level of skill and will in the searched items, and their respective specialization in the items, on a scale from one ("not important") to five ("very important"). The results illustrated in table 6.4 show that all factors are seen as nearly equally important.

Factor	1	2	3	4	5	-	Mean
Average Skill Level in Searched Items	1	3	10	16	11	0	3.80
Averageq2 Will Level in Searched Items	0	10	8	16	16	0	4.15
Specialization in Searched Items (Skill Levels)	3	5	17	12	4	0	3.22
Specialization in Searched Items (Will Levels)	1	0	11	22	5	2	3.77

Table 6.4.: The importance of the four factors included in the fitness score algorithm as estimated by 41 test subjects.

Calculating the weighting parameters w_{as}, w_{aw}, w_{ss} and w_{sw} based on the average estimations of importance of the four factors results in the following values:

Parameter	w_{as}	w_{aw}	w_{ss}	w_{sw}
Weight	25.44%	27.78%	21.55%	25.23%

Table 6.5.: Weighting parameters based on the collected data.

Comparison

The fitness score algorithm has been configured to use the aforesaid weighting parameters calculated from the data collected in the survey. Its results for the five test records have been compared to the test subjects' estimates using a two-tailed heteroscedastic T-Test with a significance level of 0.1 in order to show if the persons' estimations deviate significantly from the results calculated using the algorithm. Table C.2 lists the algorithm's result f_a, the average fitness score f_s estimated by the test subjects, the standard deviation in the collected data, and the values of the T-Test for each test row. As shown, the algorithm's results do not deviate significantly from the values collected in the survey. Using the more common significance level of 0.05, however, would show significant deviations for *Charlie* and *Donald* but the low resolution of the scale used in the survey and the small sample size do not justify such a precise analysis.

Test Record	f_a	f_s	Standard Dev.	p	$p \geq \alpha$
Alice	0.82	0.91	0.12	0.0000358436701	No
Bob	0.45	0.29	0.16	0.0000001306616949	No
Charlie	0.47	0.42	0.16	0.05912471945	No
Donald	0.5	0.44	0.17	0.05091395	No
Erika	0.19	0.08	0.18	0.0003324460113	No

Table 6.6.: Comparison of the algorithms results and the scores collected in the survey using a two-tailed heteroscedastic T-Test ($\alpha = 0.1$)

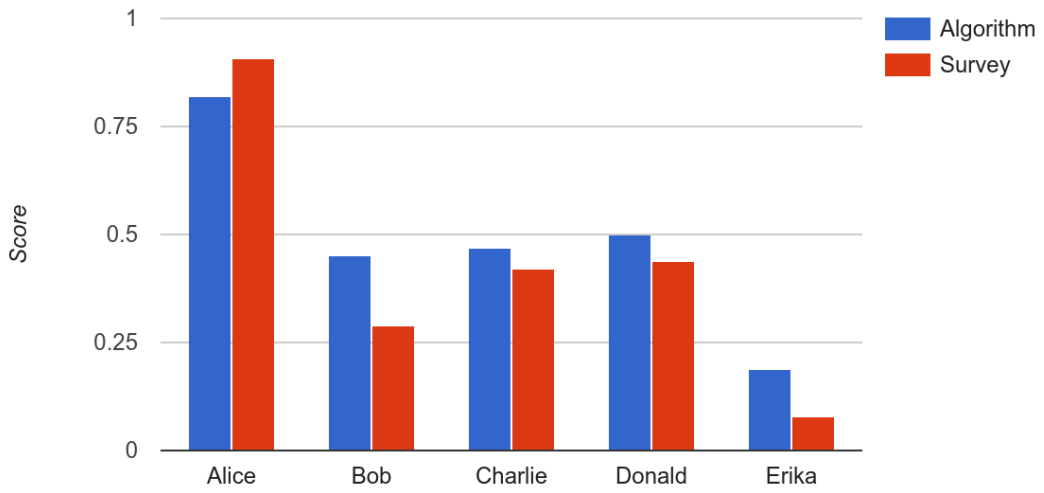

Figure 6.5.: Comparison of estimated scores collected in the survey and generated by the algorithm.

Refining the Fitness Score Algorithm

The weighting parameters generated from the survey data are all in the region of 25%; in fact, setting all parameters to 0.25 will not cause any significant change in the algorithms error rate[5], but it will result in all factors being considered equally important and thus reduce the algorithm's complexity. Furthermore, setting the factors to $w_{as} = w_{aw} = w_{ss} = w_{sw} = 0.25$ means they could be eliminated in the fitness score function[6]:

$$f = \frac{w_{as} \cdot a_s}{max(V)} + \frac{w_{aw} \cdot a_w}{max(V)} + w_{ss} \cdot s_s + w_{sw} \cdot s_w$$

$$\Rightarrow f = \frac{a_s + a_w}{4max(V)} + \frac{s_s + s_w}{4}$$

Conclusion

Comparing the algorithm's results with the values collected in the survey has shown that the introduced algorithmic approach can generally be used to generate suitable ratings of an employee's fitness into a specific set of searched skills. The analysis of the collected data has also demonstrated that the algorithm does not need to include weighting parameters since the test subjects perceive all factors to be equally important. Nonetheless, the factors will not be excluded from the algorithm's implemenation since having the possibility to tweak its working might come in handy if the future day to day use of the application reveals other requirements regarding them.

[5] It would reduce the average difference between the algorithm and the test subjects' estimations from 9.46% to 9.43% and the maximum deviation from 11.07% to 10.7%.

[6] Definitions can be found in 4.1.3.

6.2. Implementation

6.2.1. Scalability

A software system has to be able to scale according to the number of its users in order to be future-proof, as the current trend to dynamically scalable cloud solutions and server-less web architectures highlights [Mü15]. There are two concepts of preparing an application for a higher workload: vertical scaling and horizontal scaling. Vertical scaling is performed by providing more resources, e.g. memory and CPU power, to the machines running the application. Horziontal scaling, however, means setting up more machines providing the same service, so that the workload can be distributed between them [Bea]. In contrast to vertical scaling, horizontal scaling has vital advantages: the application will be more robust since the crashing of one machine can be compensated by others [Fed16], the capacity of the system can, theoretically, be unlimited, and it is cheaper because virutal machines running the service can be created dynamically if needed and then be destroyed in times of low workload, whereas the resources given to a machine that has been scaled vertically will remain unused.

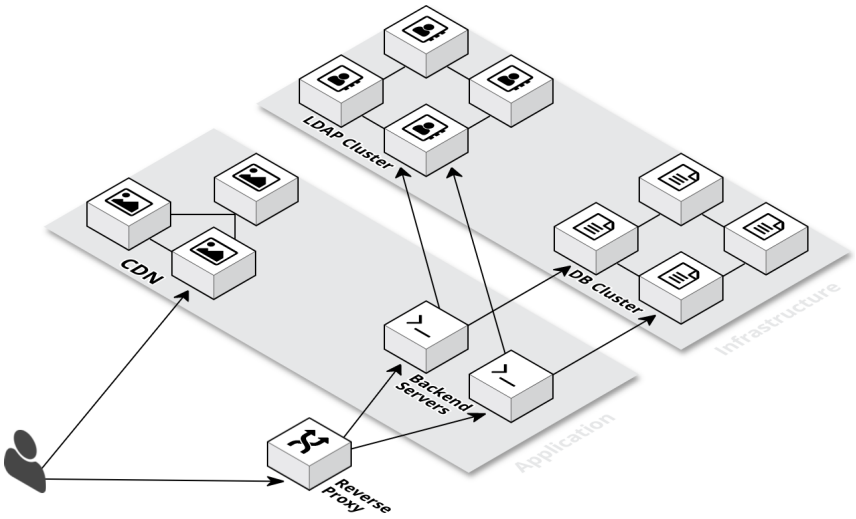

Figure 6.6.: A possible approach to scale the system using multiple backend servers, a CDN, and multiple clustered database and LDAP servers. Created with *https://cloudcraft.co*.

MongoDB

MongoDB is meant to be scaled horizontally and supports the adding of new instances to a running cluster of databases out of the box [HMPH15, p. 19], so new machines running the database as a cluster will be created if needed. As shown in figure 6.6, the backend servers can be connected to any of the database servers in order to request data. If the demanded document is not found on the instance the backend is connected to, MongoDB will handle the lookup in the cluster. To the application, the cluster is completely transparent and appears as if it was one machine.

LDAP

The LDAP servers[7] can also be run as a cluster in order to improve response times and prevent data loss by replicating the stored information [Ope]. In fact, the LDAP is currently being provided by six servers that represent the service. As illustrated in figure 6.6, the backend servers can connect to any of the LDAP servers; the data replication and synchronization are handled transparently.

Static Content

The static content like HTML, CSS and JS files, that altogether represent the frontend, are served by the reverse proxy web server. In the event of an increasing number of requests that cannot be handled by the single server, a *Content Delivery Network* (CDN) could be deployed. A CDN is a network of webservers that provide static content and large files. The reverse proxy would redirect the URLs for those files, so that the users' browsers will connect directly to said network in order to retrieve the assets.

Backend

The backend application itself does not save any data on the machine it is running on but connects to a database server (see 5.6). As a result, any number of backend instances can be set up and, in contrast to the other services, do not have to synchronize. In order to receive HTTP requests, the reverse proxy must be configured so that it redirects API calls to the backend servers. This is called *load balancing* and is supported by many modern web servers such as *nginx*[8], *Apache*[9] and *Tomcat*[10]. Tests running ten backend instances in parallel on the same machine using nginx as a load balancer[11] did not show any issues.

[7] SinnerSchrader is running *OpenLDAP* (http://www.openldap.org)
[8] https://www.nginx.com/resources/wiki/
[9] https://httpd.apache.org/
[10] http://tomcat.apache.org/
[11] Using round-robin as distribution mechanism.

Conclusion

Each component of the system has been examined for scalability; the backend software has successfully been tested in laboratory conditions, whereas the LDAP servers at SinnerSchrader already run in a scaled-up fashion. MongoDB has been designed to be horizontally scalable and comes to use in various companies like *Github*[12], *eBay*[13], and *Otto*[14], where its scalability has been proven, so that it is assumed that there will not be any problems regarding the database. Deploying a CDN that serves the static content has not been evaluated, as the implementation of the frontend is not part of this thesis, but has been worked on by Strecker [Str17]. In conclusion, the architecture can be considered as scalable beyond the needs of SinnerSchrader.

6.2.2. Response Times

As defined in 3.4.2, the application should need less than one second of response time between the user pressing the search button and the displaying of the search results. The response times of the person search API endpoint have been measured and can be found in table 6.7. The average response time of the backend is 28ms, the maximum in the test data is 44ms.

Request Parameters	Response Times (in ms)	Mean (in ms)
No parameters	26, 28, 27, 40, 27, 33, 28, 29, 26, 29	29
Specific Skill	32, 40, 30, 32, 26, 24, 28, 44, 33, 32	32
Specific Location	27, 26, 26, 25, 26, 36, 26, 22, 21, 25	26
Specific Skill and Location	36, 24, 23, 21, 22, 24, 33, 20, 30, 21	25

Table 6.7.: Measured response times of the API endpoint for the search function.

Measurements of a prototypical stage of the frontend using *Google Chrome's* built-in profiling tools showed a total response time, that is sending the HTTP request to the API, waiting for the response, parsing the response, and rendering the results, of approximately 90ms on average. The maximum response time was 106ms.

Those results demonstrate that the API is capable of serving the requests quickly enough to reach the goal of a response time under one second. The outcome of the profiling of the prototypical frontend suggest that it might even be possible to attain a total loading time of under 100ms; according to Kearney, this is would result in the users perceiving the interaction with the system as immediate [Kea17], which enhances the overall user experience.

[12]https://www.mongodb.com/presentations/mongosv-2012/mongodb-analytics-github
[13]https://www.mongodb.com/presentations/mongodb-ebay
[14]https://www.mongodb.com/industries/retail

6.3. Meeting the Requirements

In 3.4, a set of functional and non-functional requirements has been defined. The backend application that has been designed and implemented meets those requirements: the API supports the creation of user profiles that then can be retrieved, the skills in those profiles can be edited by the profile's owner only, and everybody can search for profiles of persons that offer specific skills (see table 5.1). Extra information about the employee can be added by them into their profile's comment section. New skills can be fed into the system so that people can add them to their profiles, existing skills can be edited and removed. In addition to the required features, the application offers API endpoints to recommend skills that the user might want to search for (see 4.2.2), and to recommend profiles that are similar to the one which the user is inspecting (see 4.2.3). Those features are using recommender systems to proactively propose items to the user in order to enrich the overall user experience.
The non-functional requirements include scalability, low response times, and the supported devices and browsers. As shown in 6.2.1, scalability has been partially evaluated, whereas services that the application relies on such as MongoDB and LDAP were assumed to be scalable in this architecture as comparable systems have already shown this. The response times have been evaluated in 6.2.2; the results and tests with a prototypical stage of the frontend show that the application is capable of delivering the requested information in significantly less time than defined. The requirements for supported devices and browsers, however, could not be evaluated since the determinant factor for those is the implementation of the frontend which has not been part of this thesis and will be evaluated by Strecker [Str17]. In conclusion, the backend application and the system's architecture meet all requirements.

7. Résumé and Outlook

In this thesis, the creation of a skills management application custom-tailored to SinnerSchrader has been drafted from its underlying concept to its technical design and its implementation. The motivation for SinnerSchrader to introduce such a system and the company-specific challenges it has to overcome have been outlined. To compile the requirements for the system, semi-structured interviews with representatives for the groups of stakeholders, namely the Flow Team, project managers, and *regular* employees, have been conducted. Those requirements provide the basis for the analysis of available skill management tools. As shown, two mayor factors required by SinnerSchrader cannot be served by any of the commercial solutions: the tool is supposed to put emphasis on collaboration, not supervision, and users should be able to search for persons that not only have knowledge about a certain topic but should also take into account their interests and personal preferences. These two factors form the backbone of the application's technical design; its central feature is a search function that finds the best matching employees for the searched skill set. The scoring algorithm determining how well a person matches the search query has been designed, implemented, and evaluated. Furthermore, the construction and implementation of recommender systems that enrich the user experience by proactively presenting favorable items to them have been laid out.
Both the implementation and the underlying concept have been evaluated regarding possible concerns including technical issues and the fulfilling of the end users' needs. To examine the latter, a survey has been conducted and analyzed.
Altough the application has been tested using a fair amount of generated data, live data entered by real users could invalidate the results found in the evaluation phase and reveal obstacles not considered in its design. Only a long-run test phase exposing the application to the users will provide sufficient information about those factors. Running such a test requires a graphical component to present the user with an interface; this will be subject to further research. The constructed tests, however, suggest that the novel approach of using a scoring algorithm in the context of skills management provides a solid basis to find the most suitable employee.
Although the implemented backend provides the required features, future development iterations will bring new functionality. Conceivable extensions include categories for skills, a process to manage certifications, or the possibility to customize the weighting parameters of the fitness score algorithm in the frontend.
The current state of the application, however, already shows the potential to become an effective tool for collaboration and skills management at SinnerSchrader.

A. Statement of the Works Council

Hier könnte ein Statement des BR stehen. Tut es aber nicht.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

Figure A.1.: Statement of the Works Council regarding mutual ratings.

B. Interviews

B.1. Consent Form

Hamburg, den 02.02.2017

Einverständniserklärung zur Teilnahme am Interview im Rahmen des SkillWill Tools

SkillWas?

SkillWill ist ein internes S2 Tool, das veranschaulicht, was jeder Mitarbeiter wie gut kann (Skills) und was ihn wie sehr interessiert (Wills). Die eigenen Skills und Wills werden dabei von jedem Mitarbeiter in ein Profil eingepflegt und sind dadurch für alle anderen, disziplin- und standortübergreifend zugänglich.

SkillWill schafft mehr Transparenz und gibt uns Einblicke, was hinter den Köpfen von S2 steckt. Durch dieses Wissen entstehen neue Möglichkeiten und es kann uns helfen, uns besser kennenzulernen und lose Enden zu verknüpfen.In der Einsatzplanung ermöglicht es, Mitarbeiter und ihre Talenten gezielter einzusetzen und persönliche Entwicklungswünsche besser zu berücksichtigen.Wer im Arbeitsalltag einen Ansprechpartner zu bestimmten Themen sucht, findet mit SkillWill schneller die richtige Person.

Zur evaluierung der Anforderungen an das Tool führen wir kurze semi-strukturierte Interviews durch, im Rahmen derer wir dir einige Fragen bezüglich deiner Erwartungen und Wünsche an das Tool stellen.

Nutzung der Daten außerhalb von SinnerSchrader

Die in den Interviews erhobenen Daten und die daraus gezogenen Schlüsse werden außerhalb von SinnerSchrader auch für eine Bachelorarbeit an der Uni Hamburg verwendet.

In dieser Arbeit werden zur Veranschaulichung die Antworten an Personen gebunden. Dazu soll dein Nachname genutzt werden. Natürlich kannst du dem widersprechen und wünschen, dass nur der Anfangsbuchstabe deines Namens verwendet wird, oder dass dein Name komplett pseudonymisiert wird, wodurch die Daten nicht mehr auf die zuzuordnen sind. Bitte kreuze an, welche Variante du bevorzugst:

- Ich bin mit der Nutzung meines vollständigen Nachnamens einverstanden.
- Ich wünsche, dass mein Nachname auf den Anfangsbuchstaben reduziert wird.
- Ich wünsche, dass ich komplett pseudonymisiert werde.

Erhebung von personenbezogenen Daten

Neben deinen Antworten speichern und verarbeiten wir lediglich noch deinen Nachnamen (dies trifft nicht zu, wenn du oben angegeben hast, pseudonymisiert zu werden) und deine Position zum SkillWill Tool, also z.B. ob du Projektmanager bist der er nutzen wird, um Teams aufzubauen, oder ob du bereits an der Entwicklung des Tools beteiligt warst.

Aufzeichnung des Interviews

Um die Protokollierung des Interviews zu vereinfachen, soll eine Tonaufnahme des Interviews angefertigt werden. Diese Aufnahme wird nur intern und nur zu zwecken der Protokollierung verwendet und wird nicht

Figure B.1.: Interview Consent Form (Page 1)

nach außen gegeben. Wenn du damit nicht einverstanden bist, kreuze dies bitte hier an. In diesem Fall wird keine Aufnahme angefertigt.

- ○ Ich bin mit der Anfertigung einer Tonaufnahme zu einverstanden.
- ○ Ich bin mit der Anfertigung einer Tonaufnahme nicht einverstanden.

Freiwilligkeit der Angaben

Die Teilnahme an diesen Interviews und die Abgabe von Antworten ist selbstverständlich freiwillig. Wenn du die Teilnahme verweigerst oder auf einzelne Fragen keine Antwort geben möchtest, dürfen und werden dir daraus keine persönlichen Nachteile entstehen.

Dauer und Umfang

Das Interview ist bewusst offen gestaltet, sodass die Dauer mitunter stark schwanken kann. Als grober Zeitrahmen sind 15 Minuten vorgesehen.

Einverständnis

Ich habe die oben stehenden Angaben und Bedingungen vollständig gelesen und bin mit Ihnen einverstanden. Mir ist bewusst, dass mir durch die Verweigerung der Teilnahme keine Nachteile entstehen dürfen. Ich nehme an der Umfrage aus freien Stücken teil. Ich wurde darüber informiert, dass ich jederzeit Fragen bezüglich des Interviews stellen und ohne Angabe von Gründen mein Einverständnis widerrufen kann. Mir ist bekannt, dass meine Daten nur für die interne Entwicklung des Tools und für wissenschaftliche Zwecke im Rahmen der o.g. Abschlussarbeit verwendet werden.

Name des/der Interviewten	Ort, Datum, Unterschrift des/der Interviewten
Name des/der Interviewenden	Ort, Datum, Unterschrift des/der Interviewenden

Figure B.2.: Interview Consent Form (Page 2)

B.2. Transcripts (DE)

The interviews described in 3.2 have been recorded and transcripted. The transcripts can be found in this section.

B.2.1. Mr. Gruber

Was wünscht du dir vom SkillWill Tool?

Einen guten Überblick über die Erfahrungen der Mitarbeiter, ihre Wünsche in Bezug auf gewisse Skills, auch mal zu sehen, ob sie da Lust darauf haben, oder eben auch nicht. Und daraus dann ableiten können, wohin sich die Mitarbeiter entwicklen (sollen).

Dann haben wir gleich eine Anschlussfrage: Glaubst du bei den Wills soll mit Erfasst werden wie langfristig die sind? Also ob ich jetzt nur kurz Lust habe auf Java oder auf ewig?

Ich denke mal nicht, dass wir eine Art Historie in dem Tool brauchen, sondern, dass das Tool ausreichend ist, wenn es den aktuellen Status protokolliert. Wenn sich Änderungen ergeben wird das im Tool festgehalten, aber ich brauche nicht die vorhergehenden Werte.

Aber auch keinen Blick in die Zukunft?

Den leite ich dann tatsächlich mit dem Mitarbeiter zusammen davon ab.

Skills und Wills werden auf einer numerischen Skala angegeben, wie groß sollte diese sein?

Von null bis fünf würde meiner Meinung nach ausreichen, man könnte das auch von minus zwei bis plus zwei über null aufziehen, weil der Wille ja auch negativ aussschlagend ist, da hättest du dann mit dem negativen Bereich eine Kennzeichnung, dass du da gar keine Lust drauf hast. Und das gleiche nochmal in positive ausschlagend. Null dann halt als neutrales Element.

Sollte die Gesamtberufserfahrung mit erfasst werden?

Die Gesamtberufserfahrung, nein. Besondere Merkmale hingegen ja. Zum Beispiel wenn der Mitarbeiter eine Zertifizierung für eine spezielle Software hat, Beispiel AEM, wenn da eine Schulung mitgemacht wurde, das sind Informationen, die wir durchaus gebrauchen können. Die langjährige Einschätzung, dafür sind die Vorgesetzten da, das muss nicht im Tool erfasst werden.

Zertifikate, wenn wir schon dabei sind: Als Skill erfassen, oder separat?

Guter Punkt. Das wüsste ich so auf Anhieb ehrlich gesagt nicht.

Sollte das Tool automatisch Teams erstellen können? Nach dem Motto "ich wünsche eine Team von vier Mann, die AEM, Java und was weiß ich" können und das Tool generiert dieses Team.

Würde ich nicht als Bestandteil dieses Tools sehen, weil wir für das konkrete Erfassen der Projekte ein anderes Tool verwenden. Allerdings soll dieses Tool einen Impuls geben, wer wohin gehen könnte.

Sollten Mitarbeiter einander bewerten können?

Nein.

B.2.2. Mr. Warnholz

Was wünscht du dir vom SkillWill Tool?

Erstmal würde ich davon erwarten, dass mir das in meiner Arbeit als Account Manager die Arbeit erleichtert, um passende Mitarbeiter für meine Projekte zu finden.

Die Skill- Und Willlevel werden auf einer numerischen Skala von 0 bis n angegeben. Wie groß sollte n sein, damit es sinnvoll ist? Also wie groß soll die Skala sein?

Ich würd sagen auf jeden Fall ungerade, ich würd sagen von eins bis fünf würde ausreichen.

Sollte die Gesamtberufserfahrung des Mitarbeiters in der Suche mit einbezogen werden?

Also indem man aktiv danach sucht, oder als Ausgabe dann unter Namen usw.?

Das würde mit beim Namen angezeigt und je "senioriger" der Mitarbeiter, desto weiter oben in der Suche.

Also gibts da schon eine Vorfilterung zu?

Genau das ist ja die Frage.

Also würde es dann geben. Würde ich, glaub ich, hilfreich finden, ja.

Die Will Levels: Glaubst du da soll erfasst werden, ob das ein langfristiger Will ist, oder ein kurzfristiger?

Auf jeden Fall glaube ich sollte irgendwo durch eine spätere, oder durch eine regelmäßige Abfrage oder Anpassung dieser Wills irgendwie eine Synchronisation erfolgen. Weil, das kann ja sein, dass du mal vor dreizehn Monaten angegeben hast, dass du mal AEM besser entwickeln wollen würdest, nach neun Monaten bist du dann quasi auf dem "Expert Level", aber das steht dann immernoch drin, und das willst du dann ja gar nicht mehr so dringend entwickeln oder dich weiterbilden, weil du das ja so gut kannst, aber in dem Tool steht dann immernoch drin, dass du dich da weiterentwickeln wollen würdest, das macht ja nicht so viel Sinn.

Die Idee war, dass das im Rahmen der Halbjahresgespräche gemacht wird.

Würde wahrscheinlich für die erste Zeit erstmal ausreichen.

Fändest du es sinnvoll, wenn sich Mitarbeiter untereinander bewerten?

Schwierig. Glaube nicht, als erster Impuls. Weil, da würden wahrscheinlich zu viele persönliche Befindlichkeiten mit Einfließen, also "mag ich den?", "hatte ich mit dem schonmal eine weitergehende Beziehung als die professionelle bei der Arbeit?", das kann alles da mit reinspielen. Also ich glaube da ist das schwierig

dann irgendwo abzugrenzen, was professionell ist, oder was wirklich arbeitstechnisch relevant ist, und eben was einfach nur auf Sympathie oder Nicht-Symptathie beruht.

Und last but not least: Fändest du es sinnvoll, wenn das Tool automatisch Teams zusammen stellt? Nach dem Motto: "Ich möchte ein Team von vier Leuten, die zusammen Java und AEM und PHP können?"

Why not? Könnte ich mir als sinnvoll vorstellen. Hätte ich unendlich Ressourcen, also unendlich Mitarbeiter, zur Verfügung, dann wäre es sehr wahrscheinlich, dass eine sinnvolle Zusammenstellung da rauskommt als Ergebnis. Bei der Ressourcensituation wie sie jetzt aber aktuell ist und sehr wahrscheinlich auch in Zukunft noch zugespitzter sein wird, wird das definitiv nie als Ergebnis rauskommen. Kann ich mir vorstellen. Oder bedenkt das Tool auch Verfügbarkeiten?

Nein.

Gut, dann kannst du was ich vorher gesagt habe quasi streichen. Aber das ist dann das Ding, es würde dir wahrscheinlich nach den Anforderungen, die ich eingebe, ein sinnvolles Ergebnis geben, ob die dann aber verfügbar sind, ist eine ganz andere Frage. Und das wird wahrscheinlich ein bisschen schwierig sein.

B.2.3. Ms. Spranger

Was wünscht du dir vom SKillWill Tool?

Ich wünsche mir, dass es einfach bedienbar ist, dass es viel genutzt wird, dass es einen Mehrwert bietet und die Teamplanung vereinfacht, dass es die Mitarbeiterentwicklung vereinfacht, also die Bedarfe und Wünsche besser aufzeigt und dass es uns eine bessere Übersicht verschafft.

Skill- und Willlevel werden auf einer Skala angegeben, wie groß sollte diese Skala sein?

Ich würde sagen, vier unterschiedliche Werte reichen, denn es soll ja kein Mitarbeiterbewertungstool werden wo wir auf fünfzehn unterschiedlichen Skalenschritten angeben wie gut jemand ist, damit wir Leute vergleichen können, sondern wir wollen eine ganz ungefähre Tendenz bekommen, und genauer kriegt man es sowieso nicht hin.

Sollte die Gesamtberufserfahung mit erfasst werden? Und wenn ja, sollte das in der Suche berücksichtigt werden?

Kann sicher nicht schaden, wäre die Frage, ob das zu personenbezogene Daten sind. Ob jemand Junior, Intermediate oder Senior ist, das sollte auf jeden Fall mit rein, die Anzahl an Berufsjahren, das weiß ich nicht.

Sollte das in der Suche mit berücksichtigt werden?

Wenn ich nach einem bestimmten Skill suche, dann hab ich ja die Erfahrung über das Skill Level.

Aber wenn wir jetzt die generelle Berufserfahrung betrachten und nicht auf den einzelnen Skill?

Ne, nicht immer ist mehr Jahre auch mehr Erfahrung. Deswegen nein. Da reicht es, wenn der Titel mit angezeigt wird, aber in der Suche macht das keinen Sinn.

Soll bei den Wills mit erfasst werden, ob das ein langfristiger oder ein kurzfristiger Wille ist?

Nein, sollte nicht mit erfasst werden. Wenn jetzt jemand Lust auf Java hat, dann weiß man überhaupt nicht, ob der in zwei oder drei Monaten immernoch Lust hat, und falls er dann keine Lust mehr hat, dann wird er das kund tun.

Fändest du es sinnvoll, wenn sich Mitarbeiter gegenseitig bewerten könnten?

Ein oft diskutierte Frage, aktuell nein. Man kann damit natürlich sagen, wenn man sich gegenseitig bewertet, dann ist das nicht so sehr von der Eigenwahrnehmung abhängig und das levelt sich irgendwann, auf der anderen Seite ist jegliches Bewertungssystem aus Sicht des Betriebsrats sehr kritisch zu sehen, und deswegen

glauben wir, dass durch mündliche Rücksprache und Einschätzung und so weiter ausreichend Ausgleich gegeben ist.

Möchtest du, dass das Tool in der Lage ist, Teams automatisch zusammen zu stellen?

Nein!

C. Evaluation Data

No.	v_{Ideal}	A.D.	DBs	Funk.	hybris	Kom.	MySQL	Sketch	Text	Mean Value	Min. Val.	Max. Val.	Mean Deviation	Max. Deviation
1	1.00	0.94	0.93	0.94	0.95	0.83	0.94	0.93	0.94	0.925	0.83	0.95	0.075	0.17
2	0.96875	0.93	0.75	0.84	0.94	0.82	0.93	0.93	0.76	0.8625	0.75	0.94	0.10625	0.21875
3	0.9375	0.84	0.74	0.84	0.93	0.82	0.93	0.84	0.75	0.83625	0.74	0.93	0.10125	0.1975
4	0.90625	0.84	0.64	0.83	0.93	0.75	0.93	0.83	0.75	0.8125	0.64	0.93	0.09375	0.26625
5	0.875	0.83	0.64	0.83	0.91	0.75	0.83	0.83	0.74	0.795	0.64	0.91	0.08	0.235
6	0.84375	0.83	0.62	0.83	0.83	0.74	0.83	0.75	0.72	0.76875	0.62	0.83	0.075	0.22375
7	0.8125	0.75	0.56	0.83	0.75	0.73	0.83	0.64	0.72	0.72625	0.56	0.83	0.08625	0.2525
8	0.78125	0.74	0.56	0.75	0.74	0.72	0.81	0.62	0.72	0.7075	0.56	0.81	0.07375	0.22125
9	0.75	0.65	0.56	0.75	0.73	0.72	0.7	0.62	0.64	0.67125	0.56	0.75	0.07875	0.19
10	0.71875	0.64	0.56	0.74	0.73	0.64	0.64	0.46	0.64	0.63125	0.46	0.74	0.0875	0.25875
11	0.6875	0.64	0.55	0.73	0.72	0.64	0.64	0.45	0.63	0.625	0.45	0.73	0.0625	0.2375
12	0.65625	0.56	0.54	0.73	0.72	0.63	0.63	0.45	0.63	0.61125	0.45	0.73	0.045	0.20625
13	0.625	0.56	0.54	0.73	0.64	0.62	0.62	0.43	0.62	0.595	0.43	0.73	0.03	0.195
14	0.59375	0.56	0.5	0.64	0.64	0.57	0.62	0.38	0.56	0.55875	0.38	0.64	0.035	0.21375
15	0.5625	0.55	0.44	0.64	0.64	0.56	0.56	0.37	0.45	0.52625	0.37	0.64	0.03625	0.1925
16	0.53125	0.55	0.43	0.55	0.64	0.56	0.56	0.37	0.44	0.5125	0.37	0.64	0.01875	0.16125
17	0.5	0.54	0.37	0.54	0.64	0.56	0.55	0.36	0.43	0.49875	0.36	0.64	0.00125	0.14
18	0.46875	0.46	0.37	0.53	0.62	0.55	0.55	0.35	0.37	0.475	0.35	0.62	0.00625	0.15125
19	0.4375	0.45	0.37	0.45	0.62	0.46	0.55	0.35	0.36	0.45125	0.35	0.62	0.01375	0.1825
20	0.40625	0.43	0.36	0.43	0.56	0.46	0.54	0.34	0.36	0.435	0.34	0.56	0.02875	0.15375
21	0.375	0.38	0.35	0.43	0.46	0.45	0.53	0.34	0.35	0.41125	0.34	0.53	0.03625	0.155
22	0.34375	0.37	0.27	0.37	0.35	0.35	0.53	0.27	0.35	0.3575	0.27	0.53	0.01375	0.18625
23	0.3125	0.37	0.26	0.35	0.35	0.35	0.35	0.27	0.34	0.33	0.26	0.37	0.0175	0.0575
24	0.28125	0.37	0.25	0.35	0.35	0.35	0.27	0.26	0.27	0.30875	0.25	0.37	0.0275	0.08875
25	0.25	0.35	0.25	0.35	0.34	0.35	0.26	0.26	0.27	0.30375	0.25	0.35	0.05375	0.1
26	0.21875	0.34	0.24	0.28	0.27	0.34	0.26	0.26	0.26	0.28125	0.24	0.34	0.0625	0.12125
27	0.1875	0.27	0.24	0.27	0.26	0.26	0.25	0.25	0.26	0.2575	0.24	0.27	0.07	0.0825
28	0.15625	0.25	0.24	0.26	0.25	0.26	0.24	0.25	0.25	0.25	0.24	0.26	0.09375	0.10375
29	0.125	0.17	0.24	0.24	0.24	0.25	0.24	0.24	0.25	0.23375	0.17	0.25	0.10875	0.125
30	0.09375	0.16	0.16	0.17	0.17	0.25	0.24	0.16	0.24	0.19375	0.16	0.25	0.1	0.15625
31	0.0625	0.06	0.16	0.16	0.16	0.16	0.18	0.06	0.16	0.1375	0.06	0.18	0.075	0.1175
32	0.03125	0.05	0.06	0.06	0.06	0.16	0.16	0.05	0.15	0.09375	0.05	0.16	0.0625	0.12875
33	0	0.05	0.05	0.06	0.05	0.04	0.06	0.05	0.05	0.05125	0.04	0.06	0.05125	0.06

Table C.1.: Fitness score values that have been selected to inspect the fitness scores' distribution.

Test Subject	Alice	Bob	Charlie	Donald	Erika
Subject 1	0.75	0.25	0.00	0.25	0.00
Subject 2	1.00	0.50	0.50	0.50	0.00
Subject 3	1.00	0.00	0.50	-	0.00
Subject 4	1.00	0.25	0.50	0.50	0.25
Subject 5	0.75	0.50	0.25	0.25	0.25
Subject 6	0.75	0.25	0.25	0.75	0.75
Subject 7	1.00	0.25	0.25	0.50	0.00
Subject 8	0.75	0.25	0.50	0.25	0.25
Subject 9	1.00	0.25	0.50	0.50	0.00
Subject 10	1.00	0.50	0.50	0.50	0.00
Subject 11	1.00	0.25	0.50	0.50	0.00
Subject 12	0.75	0.25	0.25	0.25	0.00
Subject 13	1.00	0.50	0.25	0.50	0.00
Subject 14	1.00	0.25	0.50	0.50	0.00
Subject 15	0.75	0.00	0.25	0.50	0.00
Subject 16	1.00	0.00	0.50	0.25	0.25
Subject 17	1.00	0.25	0.50	0.50	0.25
Subject 18	1.00	0.25	0.50	0.25	0.00
Subject 19	1.00	0.25	0.75	0.50	0.00
Subject 20	1.00	0.50	0.50	0.50	0.25
Subject 21	1.00	0.50	0.50	0.50	0.00
Subject 22	0.75	0.00	0.25	0.50	0.00
Subject 23	1.00	0.25	0.25	0.75	0.75
Subject 24	0.75	0.25	0.25	0.25	0.00
Subject 25	1.00	0.25	0.25	0.50	0.00
Subject 26	1.00	0.50	0.50	0.75	0.00
Subject 27	1.00	0.25	0.50	0.50	0.00
Subject 28	0.75	0.50	0.50	0.50	0.00
Subject 29	0.75	0.25	0.50	0.50	0.00
Subject 30	0.75	0.25	0.25	0.50	0.00
Subject 31	1.00	0.25	0.50	0.50	0.00
Subject 32	0.75	0.25	0.25	0.50	0.00
Subject 33	1.00	0.25	0.75	0.50	0.00
Subject 34	1.00	0.25	0.50	0.50	0.00
Subject 35	1.00	0.25	0.50	0.50	0.00
Subject 36	0.75	0.50	0.25	0.25	0.00
Subject 37	0.75	0.00	0.50	0.50	0.00
Subject 38	1.00	0.25	0.75	0.50	0.00
Subject 39	0.75	0.25	0.50	0.25	0.00
Subject 40	1.00	0.75	0.25	0.50	0.25
Subject 41	1.00	0.25	0.50	0.25	0.00

Table C.2.: All test subjects' responses examined in the survey.

Bibliography

[AJL+05] ARAÚJO, MLívia C. F. ; JR., Luiz H. R. S. ; LIZÁRRAGA, Miguel G. ; LING, Lee L. ; YABU-UTI, João B. T.: *User Authentication Through Typing Biometrics Features*. 2005

[Bal] BALTER, Ben: *Open source license usage on GitHub.com*. https://github.com/blog/1964-open-source-license-usage-on-github-com. – Accessed: 2017-02-14

[Bea] BEAUMONT, David: *How to explain vertical and horizontal scaling in the cloud*. https://www.ibm.com/blogs/cloud-computing/2014/04/explain-vertical-horizontal-scaling-cloud/. – Accessed: 2017-01-14

[Bec03] BECK, Simon: Skill and Competence Management as a Base of an Integrated Personnel Development (IPD) - A Pilot Project in the Putzmeister, Inc./Germany. In: *Journal of Universal Computer Science* 9 (2003), Nr. 12, S. 1381 – 1287

[Ber16] BERGMANN, Rainer: *Organisation und Projektmanagement*. 2. Aufl. 2016. Springer Gabler, 2016 (BA KOMPAKT)

[Can13] CANÓS-DARÓS, Lourdes: An algorithm to identify the most motivated employees. In: *Management Decision* 51 (2013), Nr. 4, S. 813–823. http://dx.doi.org/10.1108/00251741311326581. – DOI 10.1108/00251741311326581

[DAQ13] DARVISH, Hassan ; AHMADNIA, Hadi ; QRYSHYAN, Seyed A.: Studying the Personal Knowledge Management Profile: a Case Study at Payame Noor University. In: *Economic Insights – Trends and Challenges* 4/2013 (2013), Nr. 1, S. 1 – 12

[DMA14] DMA: *National email benchmarking report 2013*. November 2014 Edition. 2014

[DS15] DHARAVATH, Ramesh ; SINGH, Abhishek K.: Entity Resolution-Based Jaccard Similarity Coefficient for Heterogeneous Distributed Databases, 2015 (Proceedings of the Second International Conference on Computer and Communication Technologies)

[Fed16] FEDERRATH, Hannes: *Foliensatz Grundlagen der Systemsoftware SoSe 2016*. 2016

[Gut16] GUTIERREZ, Felipe: *Pro Spring Boot - A no-nonsense guide containing case studies and best practices for Spring Boot*. Apress; Springer Science+Business Media, 2016. – ISBN 978–1–4842–1431–2

[HMPH15] HOWS, David ; MEMBREY, Peter ; PLUGGE, Eelco ; HAWKINS, Tim: *The Definitive Guide to MongoDB*. 3. Apress; Springer Science+Business Media, 2015. – ISBN 978–1–4842–1183–0

[IFO15] ISINKAYE, F.O. ; FOLAJIMI, Y.O. ; OJOKOH, B.A.: Recommendation systems: Principles, methods and evaluation. In: *Egyptian Informatics Journal* 16 (2015), Nr. 3, S. 261 – 273. – ISSN 1110–8665

[IIK13] IVANOVSKA, Sashka ; IVANOVSKA, Ilinka ; KALAJDZISKI, Slobodan: Algorithms for Effective Team Building, 2013 (The 10th Conference for Informatics and Information Technology (CIIT 2013))

[Kea17] KEARNEY, Meggin: *Measure Peformance with the RAIL Model*. `https://developers.google.com/web/fundamentals/performance/rail`. Version: 2017. – Accessed: 2017-01-16

[Leh04] LEHNER, Franz: *Marktanalyse zum Angebot an Skill-Management-Systemen*. 2004

[Mas88] MASSACHUSETTS INSTITUTE OF TECHNOLOGY: *MIT License*. `https://opensource.org/licenses/MIT`. Version: 1988

[Mü15] MÜNZL, Gerald: *Cloud Computing als neue Herausforderung für Management und IT*. Berlin : Springer Vieweg, 2015

[NGI] NGINX INC.: *WHAT IS A REVERSE PROXY SERVER?* `https://www.nginx.com/resources/glossary/reverse-proxy-server/`. – Accessed: 2017-01-08

[Ope] OPENLDAP FOUNDATION, The: *Replication - OpenLDAP Administrator's Guide*. `http://www.openldap.org/doc/admin24/replication.html`. – Accessed: 2017-01-14

[PRG+17] POLLACK, Mark ; RISBERG, Thomas ; GIERKE, Oliver ; LEAU, Costin ; BRISBIN, Jon ; DARIMONT, Thomas ; STROBL, Christoph ; PALUCH, Mark: *Spring Data MongoDB - Reference Documentation*. 2017

[QS] Q-SUCCESS SOFTWARE QUALITY MANAGEMENT: *Usage of JavaScript for websites*. `https://w3techs.com/technologies/details/cp-javascript/all/all`. – Accessed: 2017-01-14

[RD00] RYAN, Richard M. ; DECI, Edward L.: Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. In: *Contemporary Educational Psychology* 25 (2000), Nr. 1, S. 54 – 67. – ISSN 0361–476X

[Sie] SIEGRIST, Kyle: *Markov Chains*. `http://www.math.uah.edu/stat/markov/Introduction.html`. – Accessed: 2017-01-13

[Sin01] SINGHAL, Amit: Modern Information Retrieval: A Brief Overview. In: *IEEE Data Eng. Bull.* 24 (2001), Nr. 4, S. 35–43

[Sin17] SINNERSCHRADER: *Quartalsbericht 1 2016/2017*. `https://sinnerschrader.ag/de/berichte/`. Version: 2017. – Accessed: 2017-01-16

[Smi15] SMITH, Ben: *Beginning JSON*. Berkeley, Calif. : Apress, 2015 (The expert's voice in web development). – 324 S.

[SSH07] SPOONAMORE, Janet H. ; SIMIEN, H. J. ; HALL, Ricky D.: *Person-to-Position Matching*. 2007

[Sta] STATISTA: *Share of users with mobile or stationary internet access in Germany in 2014*. `https://www.statista.com/statistics/432146/internet-access-devices-used-germany/`. – Accessed: 2017-01-16

[Str17] STRECKER, Katharina: *Front-End Konzeption und Umsetzung eines unternehmensinternen Webtools für Skillmanagement*. 2017

[Ueb] UEBERSAX, John S.: *Likert Scales: Dispelling the Confusion*. `http://www.john-uebersax.com/stat/likert.htm`. – Accessed: 2017-01-16

[WGGK16] WANG, Cliff ; GERDES, Ryan M. ; GUAN, Yong ; KASERA, Sneha K.: *Digital Fingerprinting*. New York, NY s.l. : Springer New York, 2016. – 189 S.

[Wil10] WILSON, Carol: *Bruce Tukman's forming, storming, norming & perfoming team development model*. 2010

List of Figures

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den _____ Unterschrift: _____

Einstellung in die Fachbereichsbibliothek

Ich bin mit einer Einstellung dieser Arbeit in den Bestand der Bibliothek des Fachbereiches Informatik an der Uni Hamburg einverstanden.

Hamburg, den _____ Unterschrift: _____

Digital Storage Device

On this device, you will find both this thesis and the codebase of the application that has been implemented as part of it. Instructions on how to build and run the application can be found in */code/README.md*.