



# Lagerverwaltungssystem

## Programmierpraktikum SS18

Robert Werner, Jialun Jiang, Daniel Wolff, Institut für Informatik  
June 25, 2018

## Outline

Motivation

User-Stories

Domänenmodell

Architekturbeschreibung

Implementationsdetails

Verwendete Technologien

Verwendete Entwicklungstools

Installationsbeschreibung

Abschließendes

## Idee

Lager Heatmap

Titel

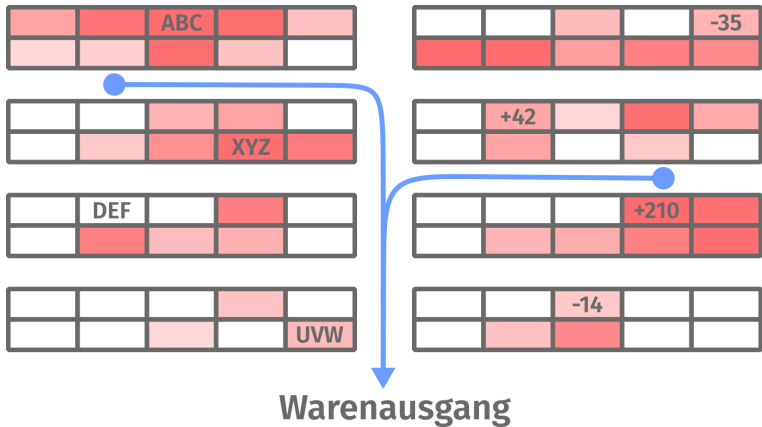
01 Bereich: Logistik

02 Person: Lagerlogistiker

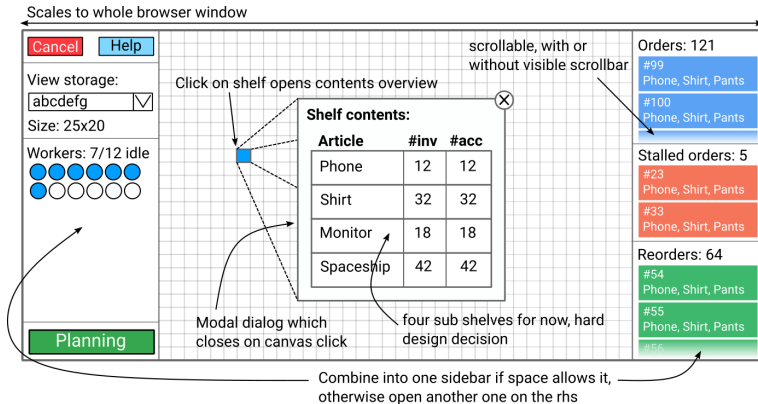
03 Situation: Möchte das Lager so  
umräumen, dass Wege minimiert  
werden

04 Problem: Weis nicht, welche Produkte im  
Lager am häufigsten gebraucht werden

## Vorstellung



## Mockup-Ansätze



## User-Stories: Regalzugriffe (1/2)

Im Detail registrieren und einsehen, welche Waren wann wo ein- und ausgehen:

### Regalzugriffe: Entnahmeübersicht

Als Lagerlogistiker will ich einsehen können, welche Lagerwaren innerhalb eines bestimmten Zeitrahmens entnommen wurden, um besser planen zu können.

### Regalzugriffe: Zugriffsabhängigkeiten

Als Lagerlogistiker will ich wissen, welche Lagerwaren innerhalb eines bestimmten Zeitrahmens zumeist gemeinsam entnommen wurden, um Abhängigkeiten zwischen diesen zu erkennen.

## User-Stories: Regalzugriffe (2/2)

### Regalzugriffe: Heatmap

Als Lagerlogistiker will ich auf einen Blick sehen, wie die Regalzugriffe im Lager räumlich verteilt sind, um ggf. Optimierungen in Auftrag geben zu können.

### Regalzugriffe: Entnahme durch Mitarbeiter

Als Lagerfachkraft will ich, dass das System mit minimalem Aufwand meinerseits meine Entnahmen registriert, sodass ich diese nicht zeitaufwendig händisch festhalten muss.

## User-Stories: Lagerübersicht

Was wird im Lager wo und in Gesamtheit gelagert?

### Lagerübersicht: Artikel einsehen

Als Lagerlogistiker will ich stets einsehen, welche Artikel in meinem Lager geführt werden, um nicht den Überblick zu verlieren oder falsch zu kalkulieren.

### Lagerübersicht: Regal einsehen

Als Lagerlogistiker will ich für jedes Lagerregal aufgelistet bekommen, was sich darin befindet, um einen besseren Überblick über den Bestand zu erhalten.

### Lagerübersicht: Lageplan für aktuelle Bestellung

Als Lagerfachkraft will ich auf einen Blick entnehmen können, in welchen Regalen sich die zur aktuellen Bestellung zugehörigen Artikel befinden, um diese effizient abzarbeiten und nicht unnötig durchs Lager zu irren.



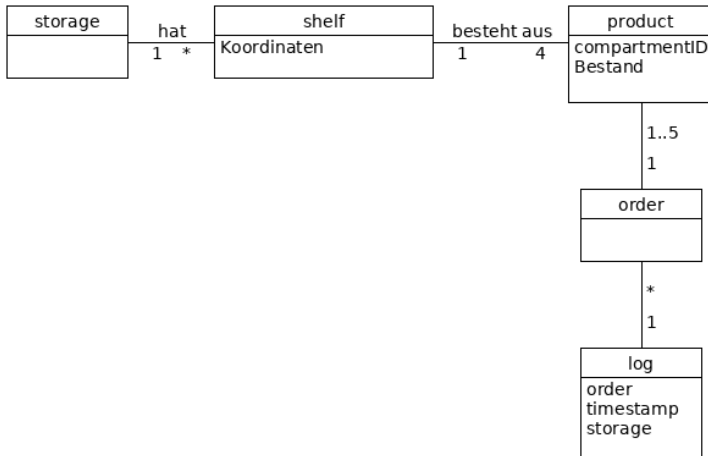
## User-Stories: Umräumung

Welche Optimierungen sind denkbar und wie gilt es, diese umzusetzen?

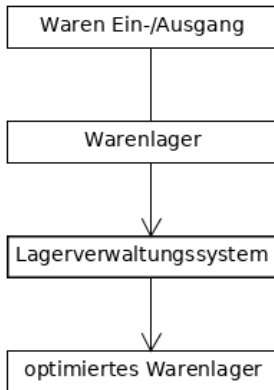
### Umräumung: Planungsvorschläge zur Wegminimierung

Als Lagerlogistiker will ich Vorschläge zu etwaigen effizienzsteigernden Umräumungen einsehen, die dabei Mitarbeiterweg, Regalzugriffe und Warenabhängigkeiten berücksichtigen, um abschätzen zu können, welche Ablaufoptimierungen sinnig sind.

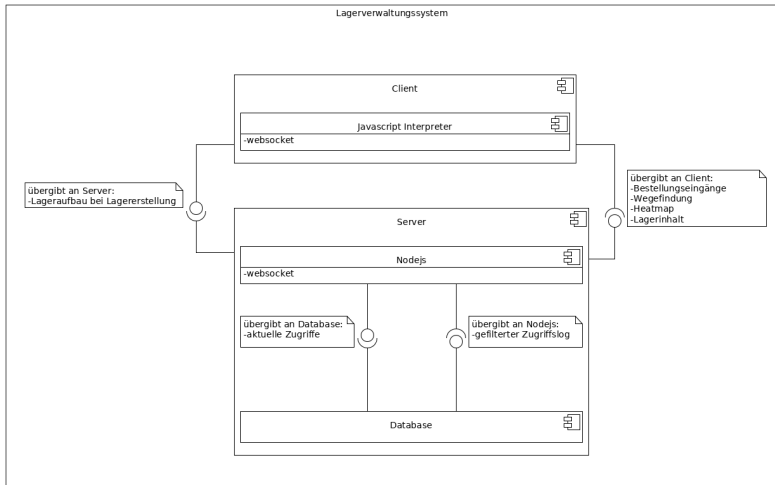
## Domänenmodell



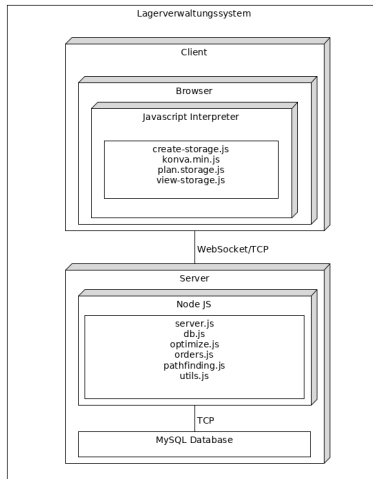
## Context View



## Structural View



## Deployment View



## Wegfindung

```
// try to find a rather efficient path for the worker to take, but not
// necessarily the shortest path possible since we're only checking
// shortest manhattan distance for the next shelf to go to and are not
// taking into account which duplication could be avoided due to one
// being along the way of another shelf. (Module: include/pathfinding.js)
exports.generateWorkerPath = (storage, order) => {
  // first step: rough super path without collision avoidance
  const closestEntrance = findClosestEntrance(storage, order);
  let path = [closestEntrance.x, closestEntrance.y];
  appendNearestShelves(path, storage, order);
  const closestExit = findClosestExit(path, storage);
  path.push(closestExit.x, closestExit.y);

  // second step: find optimal tile based sub paths between closest
  // shelves, respecting non-walkable areas.
  return getInterpolatedPath(path, storage, closestExit);
};
```

## Fächeroptimierung

```
// queries the db for a set of article ids and associated accesses over
// a given time range. We transform the original default storage into
// an optimized one by first recursively cloning it and removing all
// the subshelves within each and every regular shelf. Then we refill
// it based on the received log data by picking the closest unfilled
// shelf from each entrance until all articles are in place again.
// Access counter data will be stored within the storage to later
// visualize it in plan.js (Module: include/optimize.js)
exports.rearrangeSubShelves = (storage, fromTime, toTime, callback) => {
  db.sortedAccessesInRange(fromTime, toTime, storage._id, (results) => {
    let optimizedStorage = JSON.parse(JSON.stringify(storage));
    initAccessValues(storage, results);
    initAccessValues(optimizedStorage, results);
    const subShelves = removeAllSubShelves(optimizedStorage);
    fillSubShelvesByAccess(optimizedStorage, subShelves, results);
    calcMaxAccessCounter(storage);
    calcMaxAccessCounter(optimizedStorage);
    callback(optimizedStorage);
  });
};
```

## Technologien

NodeJS	Javascript-Runtime fürs Backend
npm	NodeJS-Package-Manager
MySQL/MariaDB	Datenbank für Artikel und Zugriffs-Log
WebSocket	Client-Server-Kommunikation
Konva	HTML5-Canvas-Abstraktion
NoUiSlider	Multifunktionsschieberegler

Ansonsten HTML5, CSS, ECMAScript 2017.



## npm-Module

http+ws	Client-Server-Kommunikation via Websocket
express	HTML-Seiten an Clients ausliefern
fs	plattformübergreifende Filesystem-Abstraktion
mysql	MySQL/MariaDB-JS-Wrapper

## Entwicklungstools

Bash+Nodemon	kleinere Skripte zu Automatisierungszwecken
Browser-Konsolen	Logging, Debugging, Profiling
Git+GitHub	Quelltext-Versionskontrolle
GitHub Projects	Kanban-lite zur Aufgabenaufteilung
GitHub Issues	Bugtracking und Verbesserungsvorschläge
GitHub Wiki	User-Stories, Mockups, Dokumentation, Notizen
Telegram	Gruppenchat, Absprachen
Inkscape	Vektorzeichenprogramm für Mockups, Grafiken

## Installation (ausführlicher auf GitHub)

- Projekt von GitHub clonen
- node, npm und mariadb installieren
- `npm install` ausführen, um Abhängigkeiten herunterzuladen

## Datenbank-Initialisierung via Commandline:

```
systemctl start mysql.service  
mysql -u root
```

```
create database programmierpraktikum;  
use programmierpraktikum;  
create user 'programmierpraktikum'@'localhost'  
    identified by 'aasfayzpu8pvleff';  
grant all privileges on programmierpraktikum.* to  
    'programmierpraktikum'@'localhost' with grant option;  
source datenbankmodell/programmierpraktikum.sql;
```

## Nutzung als Entwickler

- `./server.sh` im Quellverzeichnis ausführen; startet DB und Server
- Im Browser `localhost:8080` aufrufen
- **Create Storage** zum Anlegen eines Lagers und anschließender Befüllung
- **View Storage** erlaubt Live-Ansicht eines bereits erstellten Lagers
- **Plan Storage** führt zu Optimierungseinstellungen hinsichtlich Lageraufbau

Server kann mit mehreren Clients und Lagern gleichzeitig umgehen, Client betrachtet hingegen immer nur eines.



## Lessons learned

TODO: necessary?



**Et voilà!**

**Vielen Dank für die Aufmerksamkeit!**  
**Fragen, Anmerkungen?**