

Überblick

- Einleitung
- Zeichentool Implementierung
 - Systemarchitektur
 - Frontend
 - Backend
 - Build und Deployment
- Software demo

Einleitung

Einleitung

- Einleitung
- Kanban-Workflow
- Motivation
- Idee

Einleitung: Teamvorstellung

Team

- Alexej Esau
- Ruben Gees
- Karsten Tymann
- Oxana Zhurakovskaya

Einleitung: Teamvorstellung

Ziel

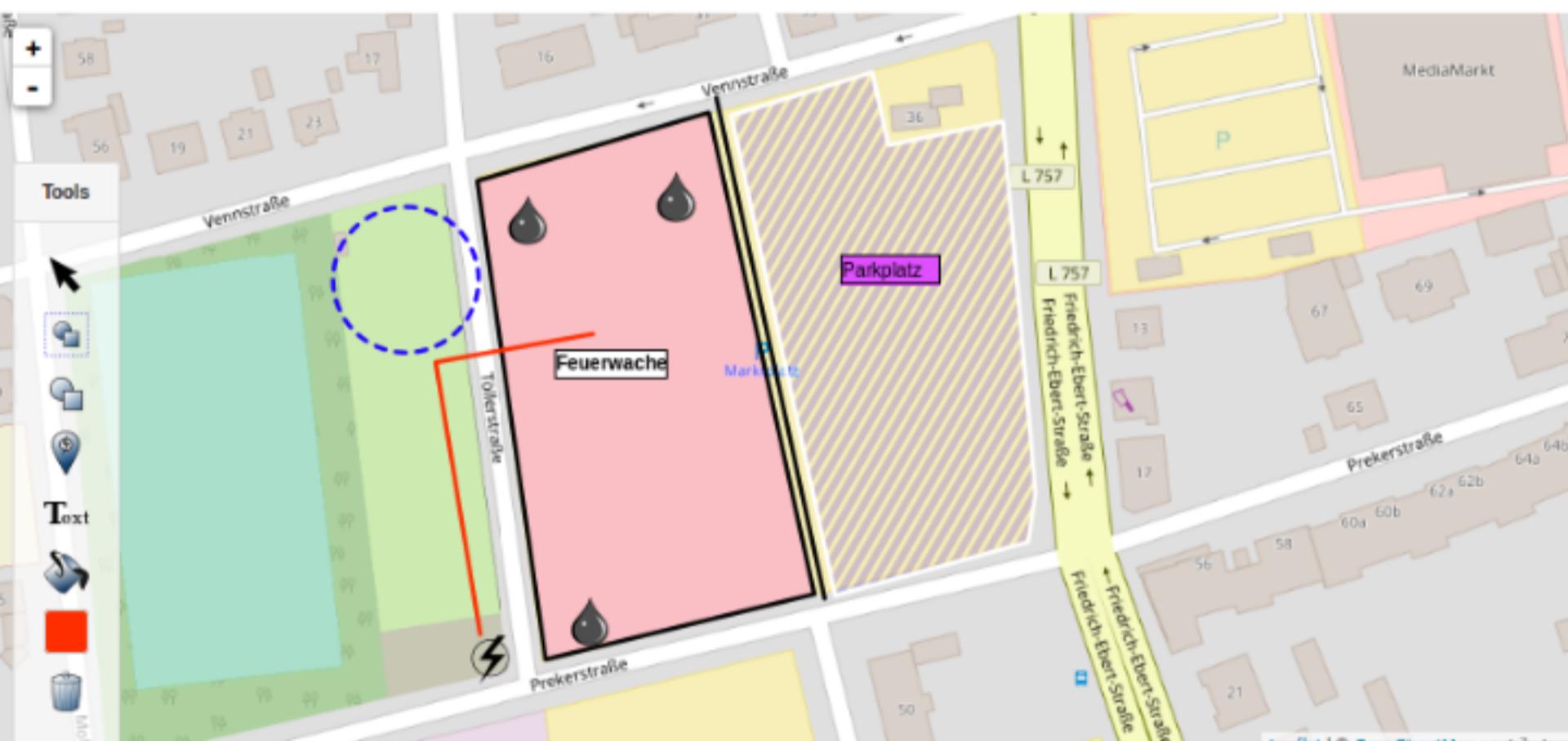
Ein einfaches grafisches Raumplanungstool



Übersicht

Einleitung: Teamvorstellung

Räumlicher Gestaltungsbereich in der interaktiven Karten



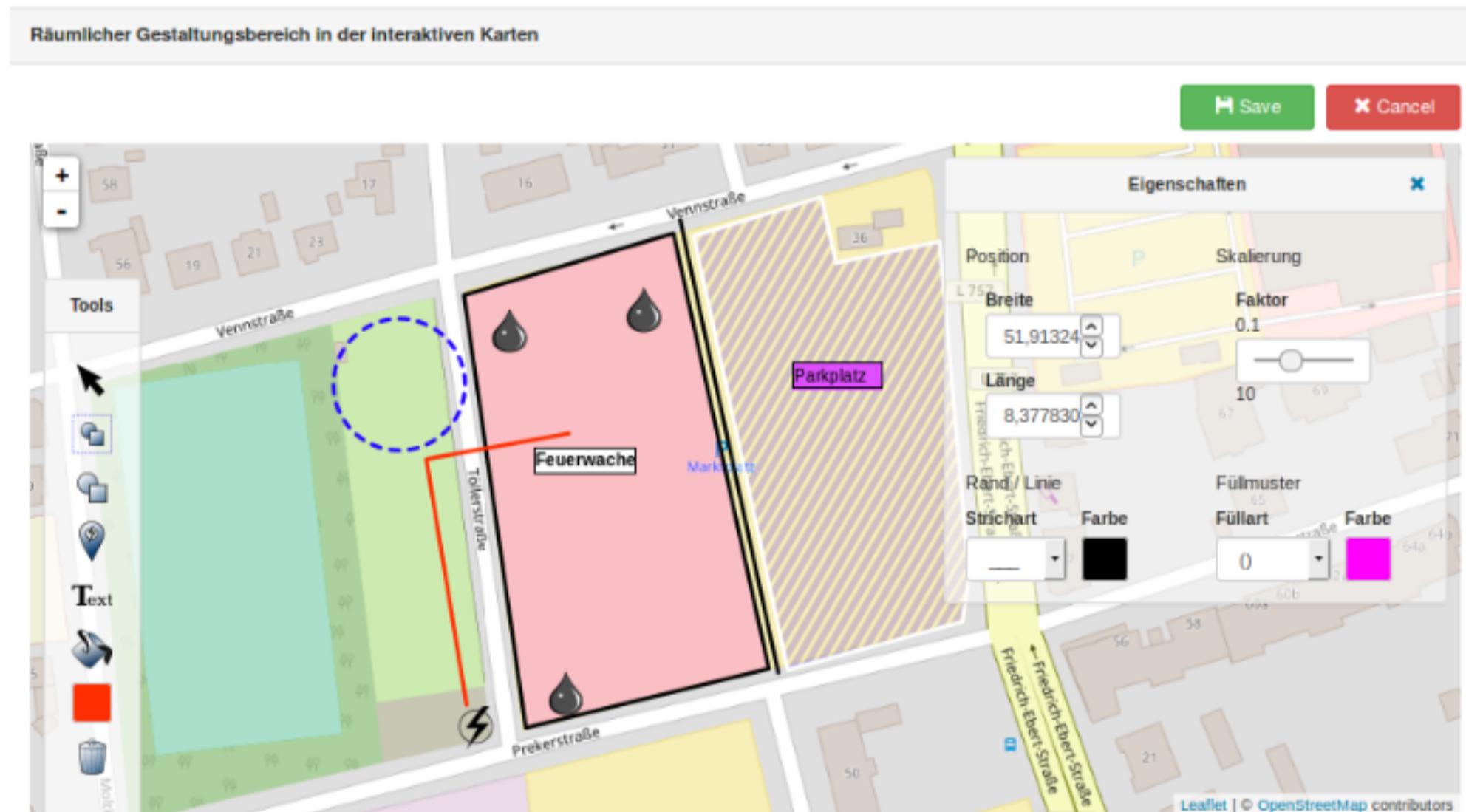
Tools

Save Cancel

Klicken Sie auf ein Element um es auszuwählen.

Leaflet | © OpenStreetMap contributors

Einleitung: Teamvorstellung



Einleitung: Teamvorstellung

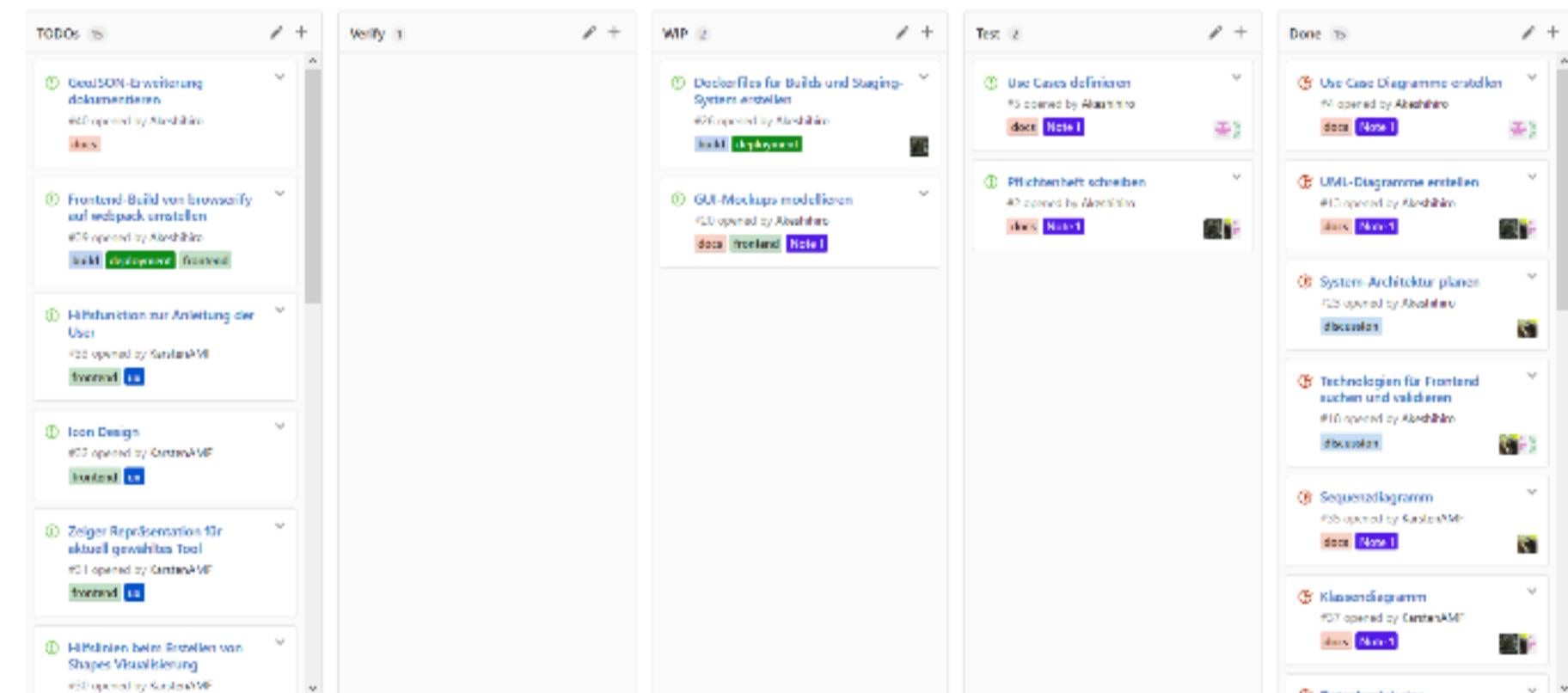
Rollenverteilung

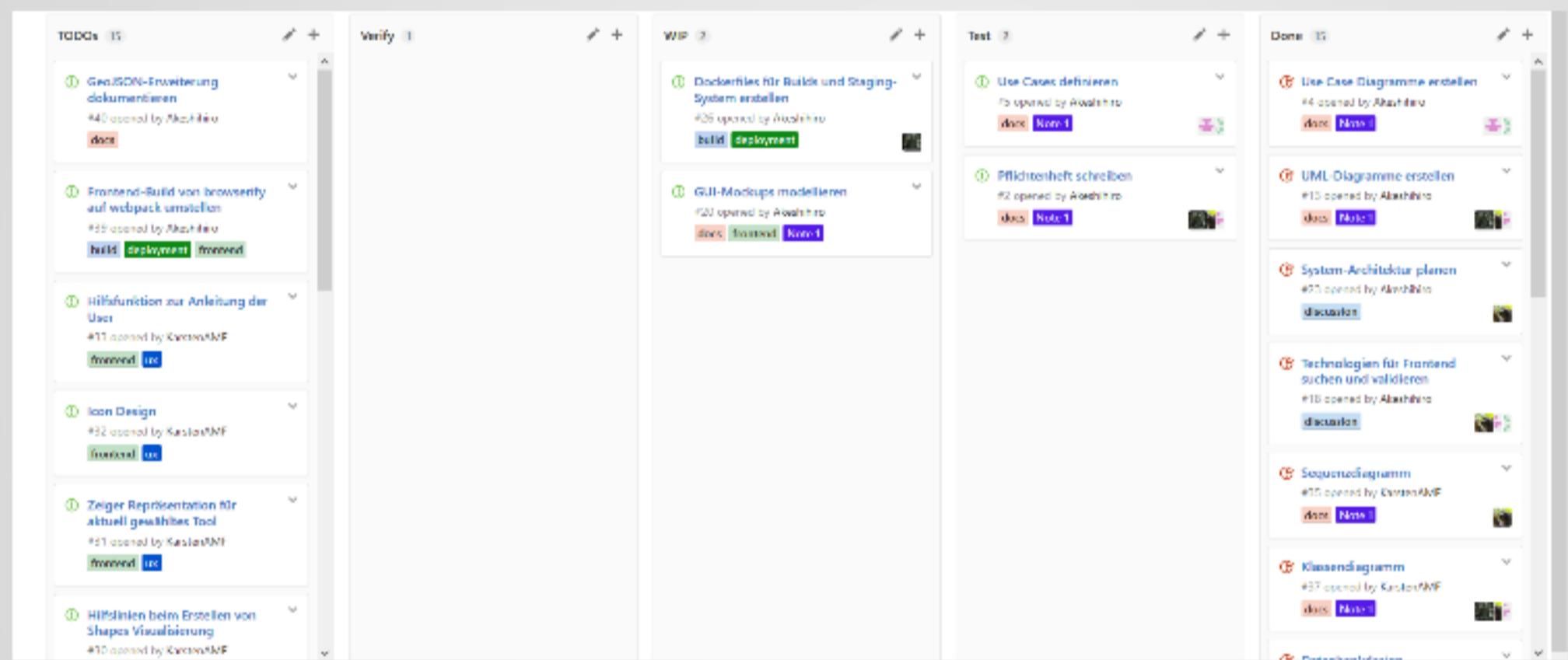
Rolle/Aufgabenbereich	Personen
Frontend-Entwickler	Karsten, Oxana
Backend-Entwickler	Alexej, Ruben
DevOps	Alexej
Tester	Alle
Dokumentation	Alle
Teamleiter	-

Einleitung: Teamvorstellung

Kanban-Workflow (vereinfacht)

- Pull-Prinzip: Aufgaben sich selber erteilen/erzeugen
- => Selbstorganisation
- Übersicht wer an was arbeitet

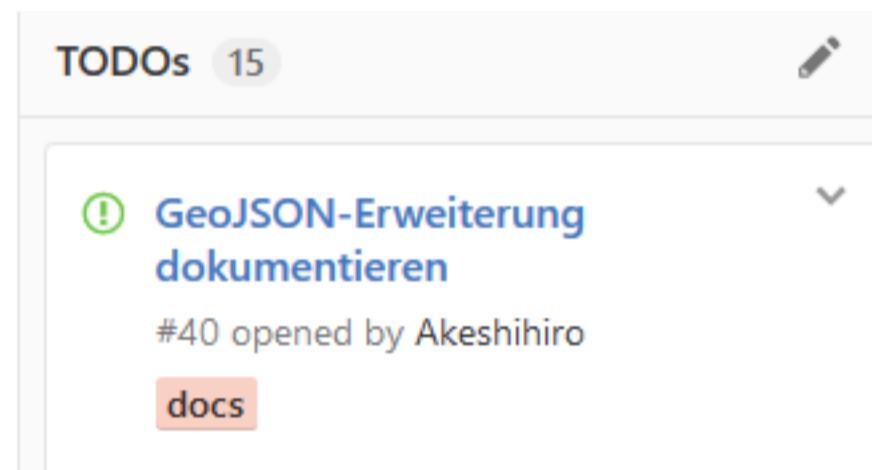




Einleitung: Teamvorstellung

Stage 0: TODO

- Anstehende Aufgaben
- Aufgaben zuteilen und erzeugen



Einleitung: Teamvorstellung

Stage 1: VERIFY

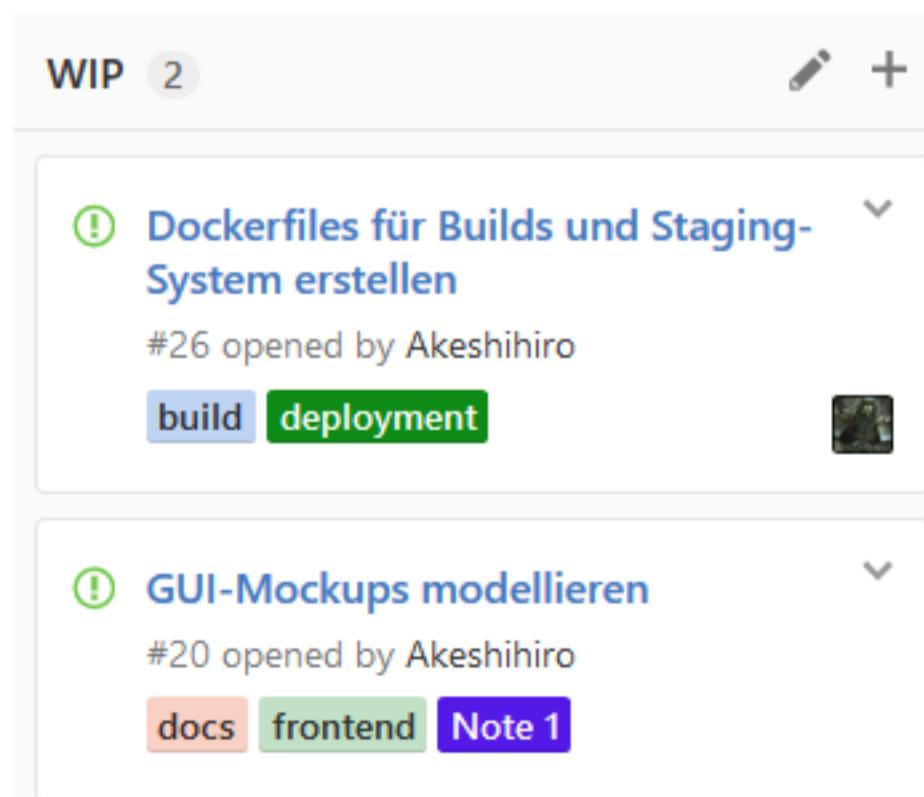
- Task auf Bearbeitung vorbereiten
- Informationen sammeln und Planung



Einleitung: Teamvorstellung

Stage 2: WIP (Work in Progress)

- Task die aktuell bearbeitet werden
- Arbeit protokollieren, Diskussion, abschließen



The screenshot shows a digital workspace interface for managing work in progress. At the top, a header bar displays "WIP" and the number "2". Below the header are two task cards:

- Dockerfiles für Builds und Staging-System erstellen**
Status: Opened by Akeshihiro #26
Labels: build, deployment
- GUI-Mockups modellieren**
Status: Opened by Akeshihiro #20
Labels: docs, frontend, Note 1

Einleitung: Teamvorstellung

Stage 3: Test

- Testen

Test 2

[Edit](#) [+](#)

① Use Cases definieren
#5 opened by Akeshihiro

[docs](#) [Note 1](#) 

① Pflichtenheft schreiben
#2 opened by Akeshihiro

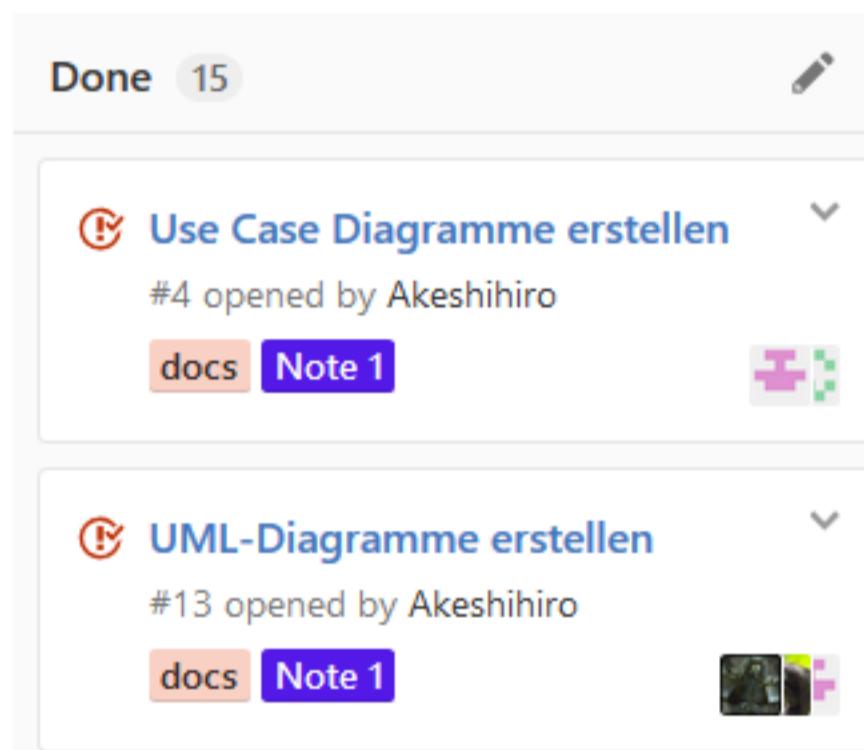
[docs](#) [Note 1](#) 

Einleitung: Teamvorstellung

Stage 4: Done

- abgeschlossene Arbeiten

Done 15



The screenshot shows a digital task board interface with a header 'Done 15'. Below, there are two completed tasks listed:

- Use Case Diagramme erstellen**
#4 opened by Akeshihiro
docs Note 1 
- UML-Diagramme erstellen**
#13 opened by Akeshihiro
docs Note 1 



Einleitung: Teamvorstellung

dahinter stecken Issues

für Diskussionen, Hilfe, interne Protokollierung

Klassendiagramm #37

Closed KoenenMF opened this issue 4 days ago · 0 comments

KoenenMF commented 4 days ago
Anhänger für preiswert

KoenenMF assigned AllesBlaue, rubengass, ronanw, and KoenenMF 4 days ago

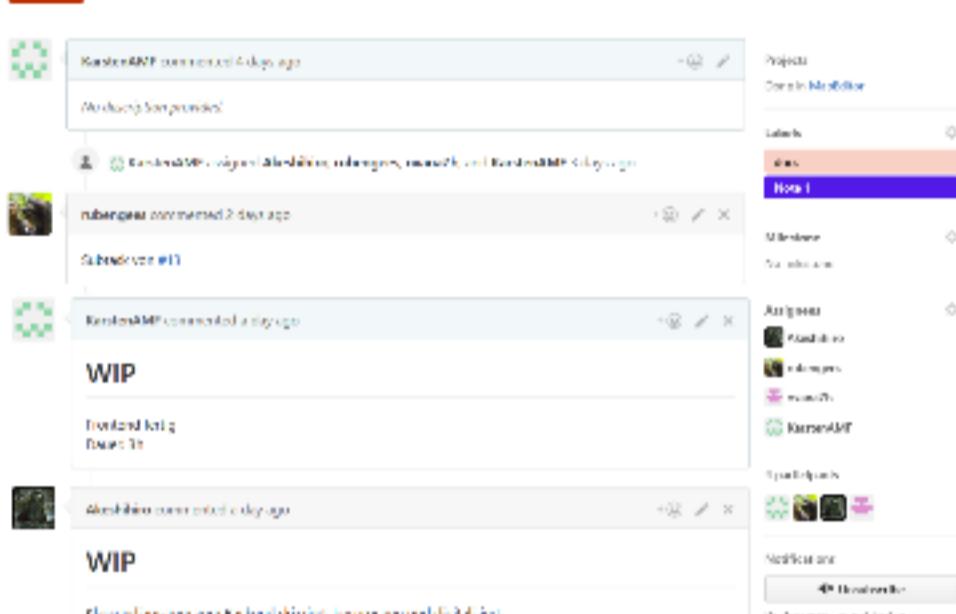
rubengass commented 2 days ago
Subscribed to it

KoenenMF commented 1 day ago
WIP

Frontend backlog
Based: Th

AllesBlaue commented 1 day ago
WIP

classDiagram comment for KoenenMF, KoenenMF, ronanw, and AllesBlaue
Based: Th



Icon Design #32

Open KoenenMF opened this issue 7 days ago · 1 comment

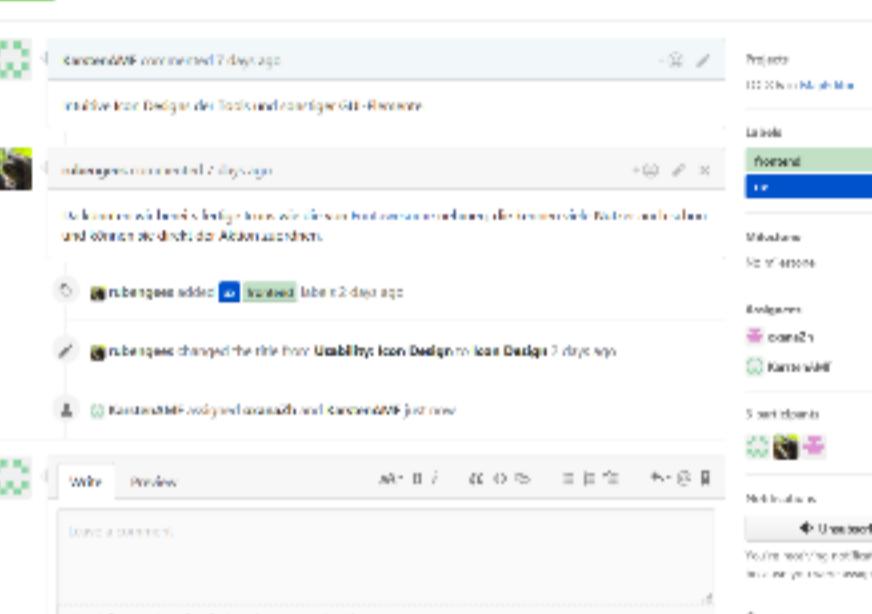
KoenenMF commented 7 days ago
Initiative Icon Designer der Tools und sonstige GitHub-Resourcen

rubengass commented 7 days ago
Ich kann es nicht leicht festlegen wie dieses Icon aussehen sollte, da es verschiedene Nutzer und Funktionen umfasst. Ich denke es sollte die Abstraktion haben.

rubengass added WIP label 2 days ago

rubengass changed the title from **Usability Icon Design** to **Icon Design** 2 days ago

KoenenMF resolved ronanw and ronanw just now



Einleitung: Teamvorstellung

Assignees

(sich selber) einer Aufgabe zuweisen

Assignees 

Assign up to 10 people to this issue 

Filter people

Clear assignees

KarstenAMF

oxanaZh

 Akeshihiro

 rubengees Ruben Gees



Assignees

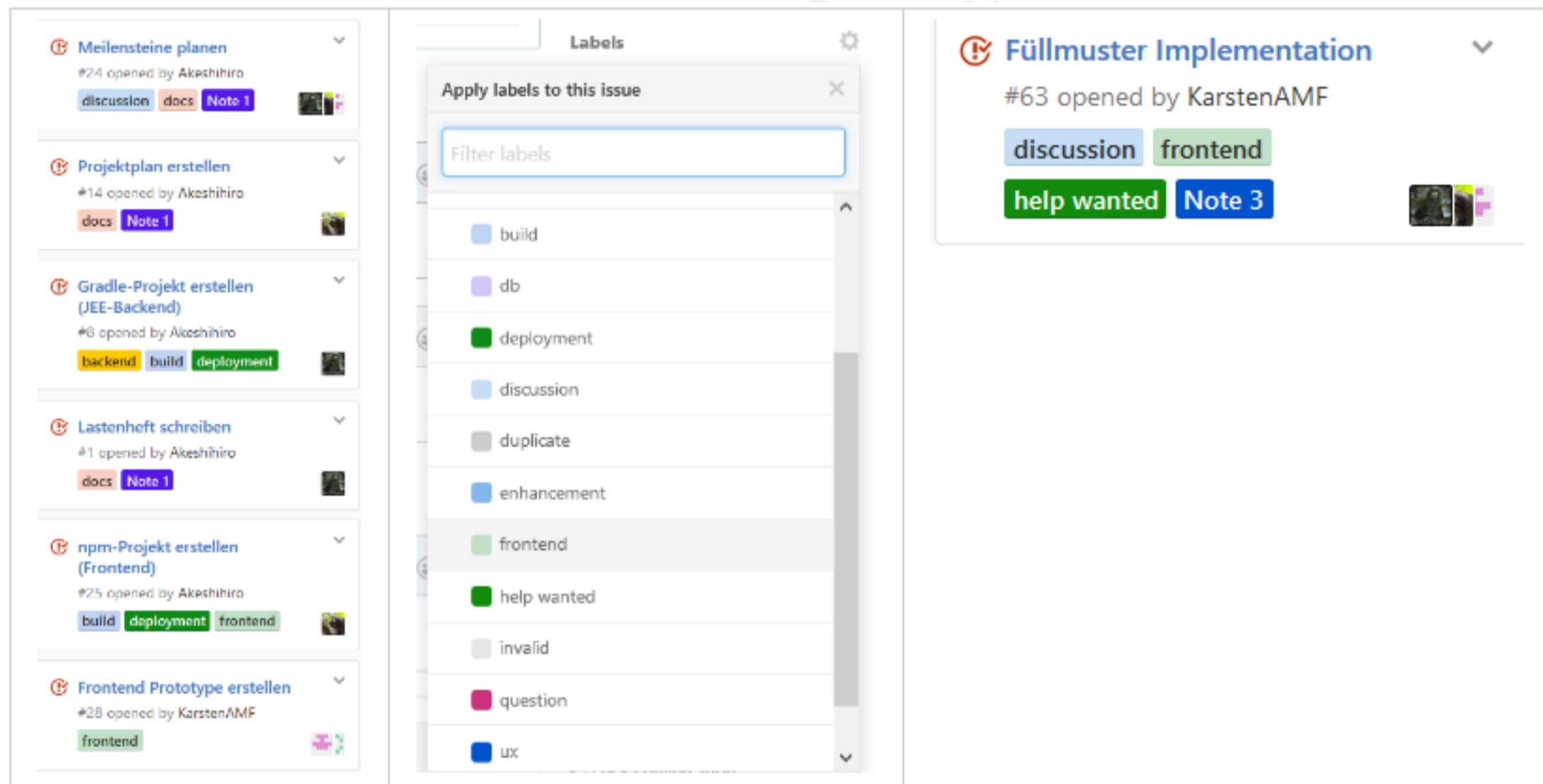
 oxanaZh

 KarstenAMF

Einleitung: Teamvorstellung

Labels

strukturieren und beschreiben Aufgabentyp



The screenshot displays a user interface for managing tasks and issues. On the left, a list of tasks is shown, each with a title, a brief description, and a list of applied labels. In the center, a modal window titled "Labels" shows a list of available labels with color-coded squares next to them. On the right, a detailed view of an issue is presented, including its title, creation date, assignee, labels, and a small preview image.

Task Title	Description	Labels
Meilensteine planen	#24 opened by Akeshihiro	discussion, docs, Note 1
Projektplan erstellen	#14 opened by Akeshihiro	docs, Note 1
Gradle-Projekt erstellen (JEE-Backend)	#8 opened by Akeshihiro	backend, build, deployment
Lastenheft schreiben	#1 opened by Akeshihiro	docs, Note 1
npm-Projekt erstellen (Frontend)	#25 opened by Akeshihiro	build, deployment, frontend
Frontend Prototype erstellen	#28 opened by KarstenAMF	frontend

Labels

- build
- db
- deployment
- discussion
- duplicate
- enhancement
- frontend
- help wanted
- invalid
- question
- ux

Füllmuster Implementation
#63 opened by KarstenAMF

discussion, frontend, help wanted, Note 3

Einleitung: Motivation

Motivation

Raumplanungstool für und mit Tetraeder



Einleitung: Motivation

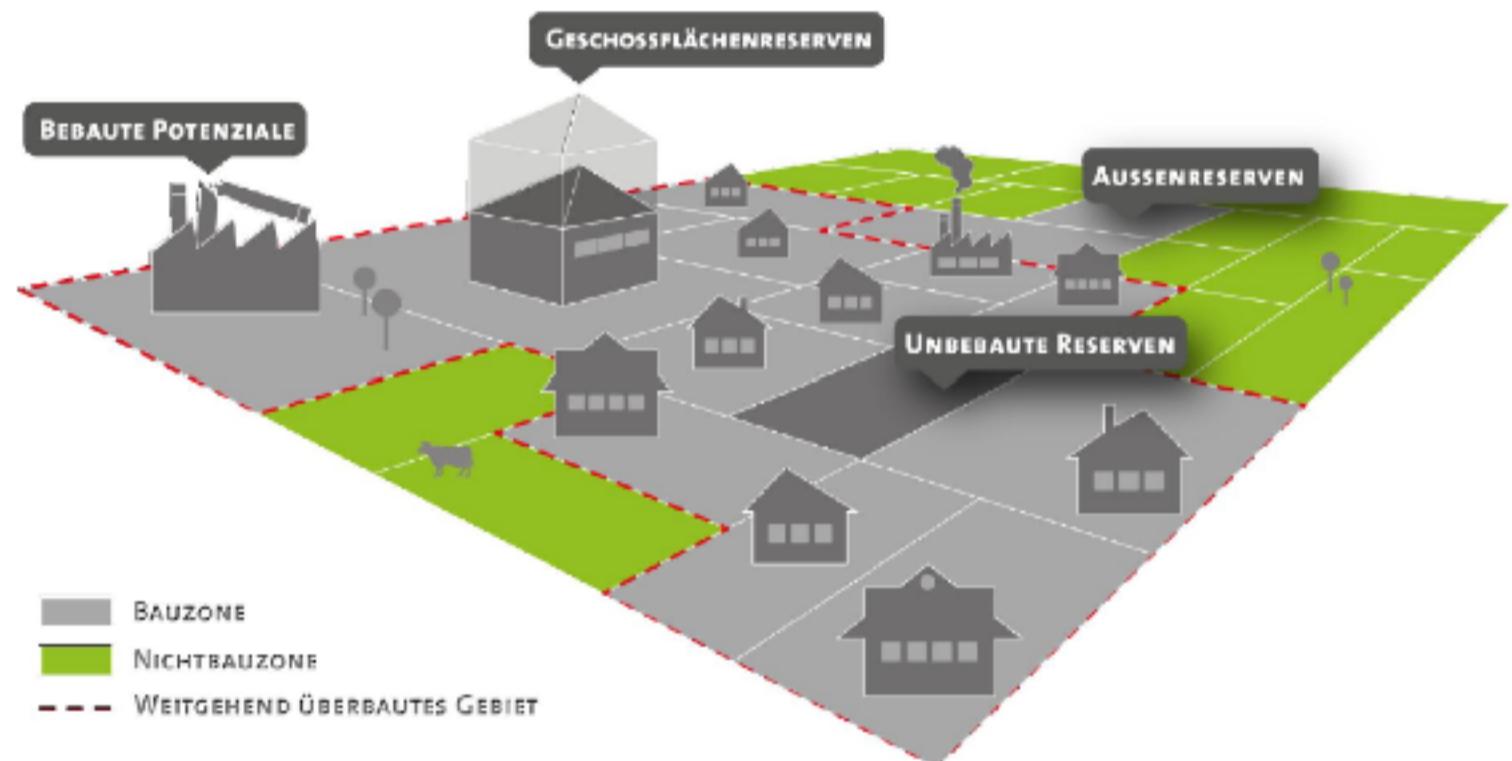
Raumplanung



Unter Raumplanung werden die planerischen Vorgänge subsumiert, einen geographischen Raum, [...] nach seinen naturräumlichen, wirtschaftlichen und sozialen Möglichkeiten zu ordnen[...]

Einleitung: Motivation

- Raum eingrenzen
- Raum ordnen
- Textliche Stellungnahme
- Raum grafisch aufarbeiten



Einleitung: Motivation

Beispiel: Marktplatz Gütersloh



Einleitung: Motivation

Alte Variante

- Kartennausschnitt als Bild
- Stellungnahme über Text

Allgemeine Daten

Planungsträger: Nordrhein-Westfalen
Planverfahren: 00_Ka_Demo2 TEST44
Verfahrensschritt: Beteiligung der Regionalplanungsbehörde im Rahmen der landesplanerischen Anpassung
Verfahrensart: Bebauungsplanung
Laufzeit: 04.10.2016 bis 20.10.2016

Stellungnahme der Behörde: Stellungnahme wurde nicht bearbeitet

Einfachender Text:

Einf. Text

Besondere Hinweise zur aktuellen Beteiligung:

An dieser Stelle haben Sie die Möglichkeit, zu der Planung Stellung zu nehmen und diese Stellungnahme mit den dazugehörigen Dateien zu übermitteln.

Räumlicher Geltungsbereich in der interaktiven Karte



Legende

Hintergrundkarten

- OpenStreetMap
- Luftbild (RVR)
- DOKS

Folien

- Bebauungspläne
- Sitzungen
- Flächennutzungspläne
- Stadtteile

Hinweis:

Nr. dem "+" Zoom oder durch das Scrollen mit der Maus, werden die Pläne sichtbar. Mit einem Klick auf einen Plan

zur Beteiligung aufgefordert

aufgefordert durch:
tetraedron: Stadt: Demo
Abgabefrist: 29.10.2016

» [Stellungnahme verfassen](#)

Bezug zum Plan

- » 103/6 Battentriple-Siedlung
- » 17.05.2016: Kaseme
- » [meine Beteiligung](#)
- » [Ganz neuer Testplan](#)

Interne Vorhabendetails

» [Interne Beiträge / Kommentare](#)

Umlauf

» [exportieren](#)

Plandaten

- » [Allgemeine Daten](#)
- » [Verfahrensschritte](#)
- » [Plandaten](#)
- » [Plandarstellung](#)
- » [test](#)

Weitere Daten für Behörden

» [Daten](#)

» [Anschriften als PDF](#)

Einleitung: Motivation

Alte Variante

Stellungnahme vorbereiten

[\[zurück zu den Plandaten\]](#)

■ Stellungnahme

Planungsträger: Nordrhein-Westfalen

Planverfahren: 00_Ka_Demo2 TEST44

Verfahrensschritt: Beteiligung der Regionalplanungsbehörde im Rahmen der landesplanerischen Anpassung

Verfahrensart: Bebauungsplanung

Laufzeit: 04.10.2016 bis 29.10.2016

Stellungnahme der Behörde: Stellungnahme wurde nicht bearbeitet

Ihr

Aktenzeichen:

Textvorlagen:

Text:^{*}

■ Anhang

Einleitung: Idee

Idee

Altes Tool:

- nicht interaktiv
- umständlich
- Plan ohne Bebilderung schwierig nachzuvollziehen

Schwerpunkte beim eigenen Tool

Alles was in der Raumplanung mit Stift und Papier gezeichnet werden kann soll das Tool ermöglichen

Einleitung: Idee

Alles was in der Raumplanung mit Stift und Papier gezeichnet werden kann soll das Tool ermöglichen

- **Hohe Usability** - Sachbearbeiter sind unerfahren und technisch unversiert
- **Funktionalität** - Kreise, Linien, Formen - wie auf Papier
- **Verlauf** - Änderungsverlauf speichern und zeigen

Zeichentool Implementierung

- Überblick
- Systemarchitektur
- Frontend
- Backend
- Build und Deployment

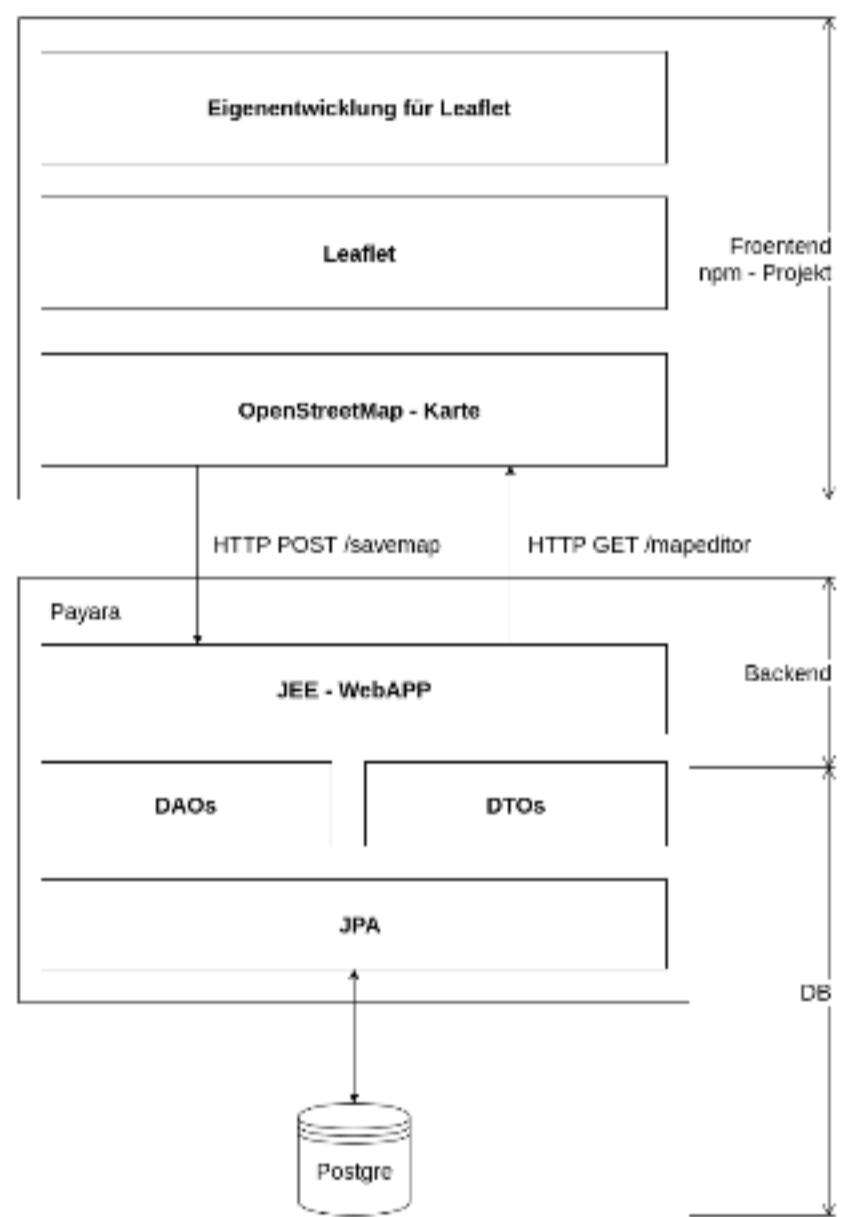
Implementierung: Überblick

Verwendete Technologien



Implementierung: Überblick

Systemarchitektur



Frontend

- Leaflet
- Usecase Diagramme
- Grundfunktionen
- Codestruktur
- ECMA Script 6
- Babel

Implementierung: Frontend

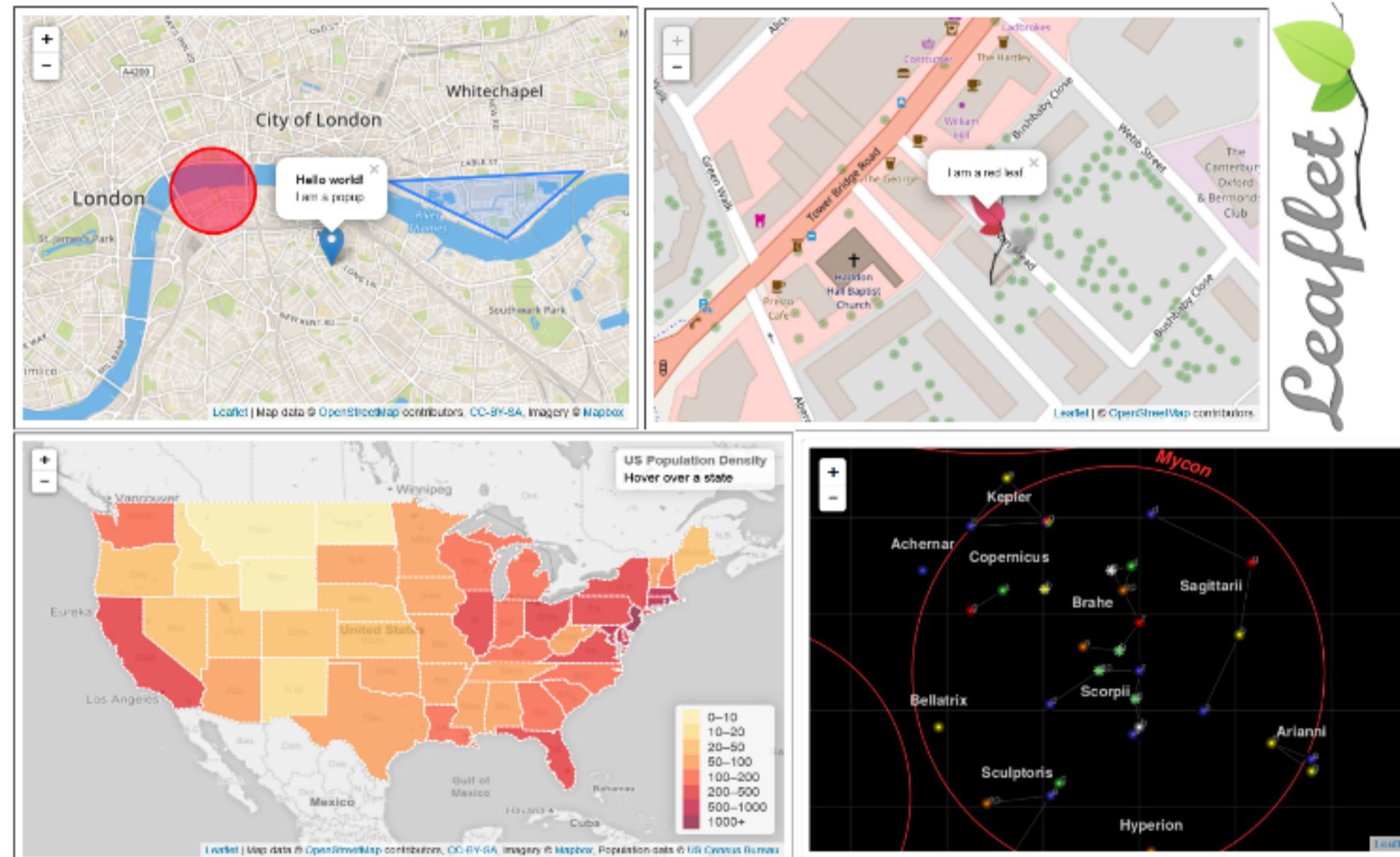


OpenSource JS-Library für interaktive Karten

- Aktuelle Version: 1.0.2 (Nov21,2016)
- Unterstützt alle gängigen Browser
- Mobile Friendly
- Exportiert Shapes in GeoJSONs
- Erweiterbar durch Plugins

Implementierung: Frontend

Beispiele



Leaflet

Implementierung: Frontend

Leaflet Basics

- Karte erzeugen
- `var map = L.map("me-map");`
- View (Koordinaten) und Zoom-Level setzen
- `map.setView([51.505, -0.09], 18);`
- Tile-Layer (Karte) setzen
 - `L.tileLayer("http://{s}.tile.osm.org/{z}/{x}/{y}.png", {
attribution: '© OpenStreetMap contributors',
maxZoom: 19
}).addTo(map);`

Implementierung: Frontend

- Event-Listener hinzufügen

```
function onMapClick(e) {  
    alert("You clicked the map at " + e.latlng);  
}
```

- map.on('click', onMapClick);

- Shape hinzufügen

```
var circle = L.circle([51.508, -0.11], {  
    color: 'red',  
    fillColor: '#f03',  
    fillOpacity: 0.5,  
    radius: 500  
});addTo(map);
```

- Layer Groups

```
shapesLayer = L.layerGroup();
```

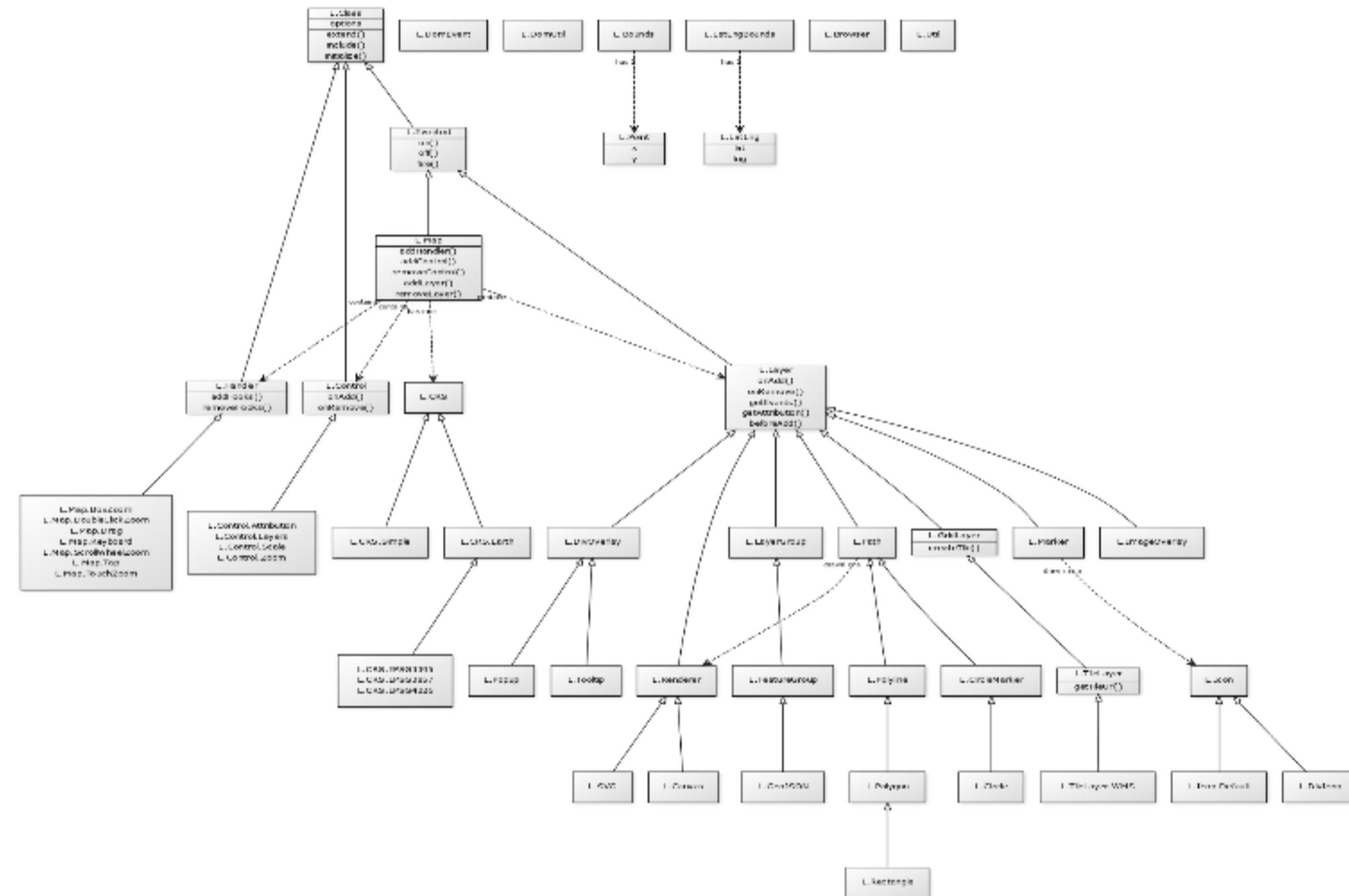
- shapesLayer.addTo(map);

Implementierung: Frontend

- **Plugin Bibliothek**
Leaflet Plugins

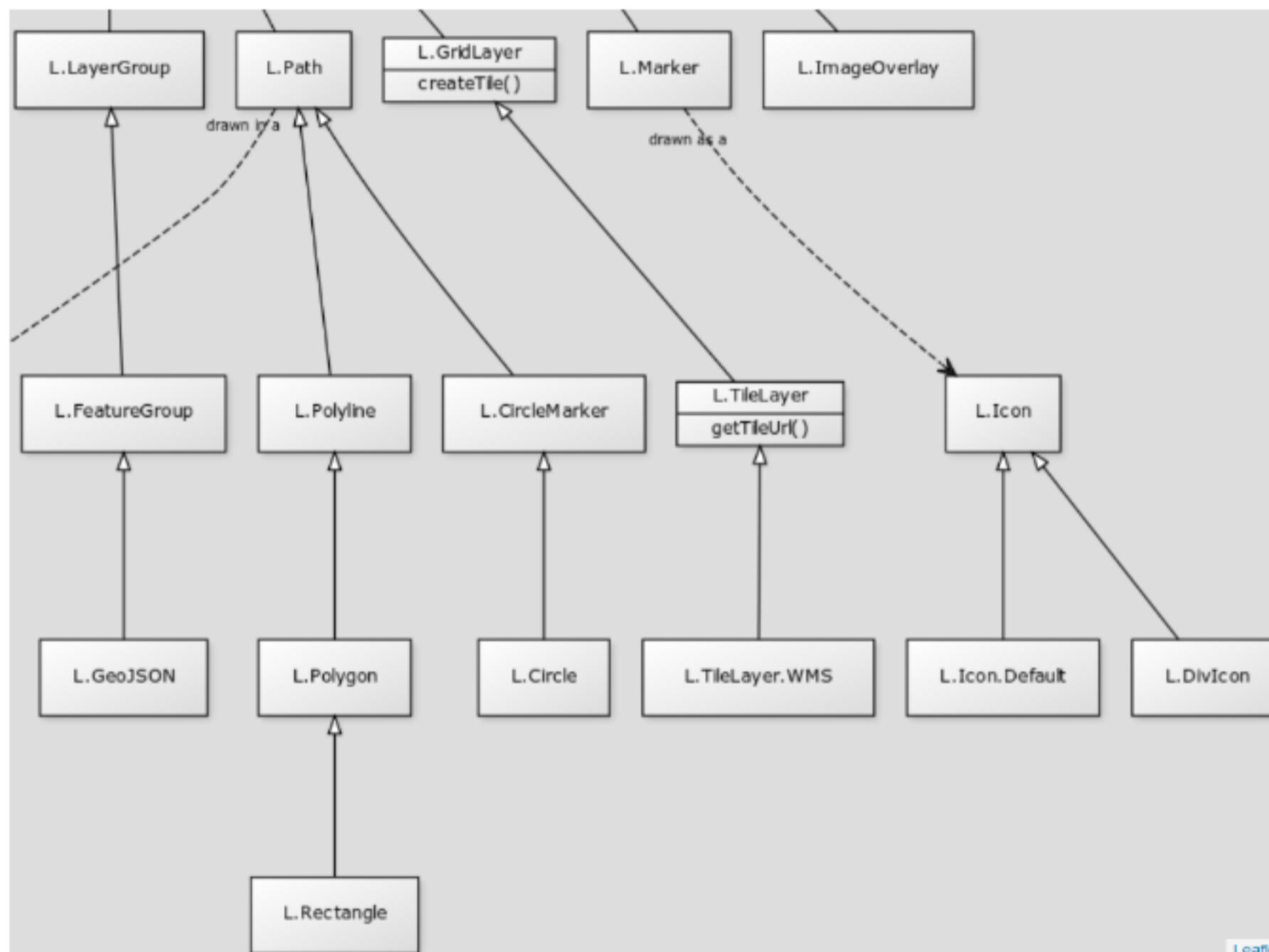
Tile & image layers	Overlay Display	Map interaction	Miscellaneous
Basemap providers	Markers & renderers	Layer switching controls	Geoprocessing
Basemap formats	Overlay animations	Interactive pan/zoom	Routing
Non-map base layers	Clustering/decluttering	Bookmarked pan/zoom	Geocoding
Tile/image display	Heatmaps	Fullscreen	Plugin collections
Tile load	DataViz	Minimaps & synced maps	
Vector tiles		Measurement	Integration
		Mouse coordinates	
Overlay data	Overlay interaction	Events	Frameworks & build systems
	Edit geometries	User interface	3rd party
Overlay data formats	Time & elevation	Print/export	
Dynamic data loading	Search & popups	Geolocation	
Synthetic overlays	Area/overlay selection		Develop your own
Data providers			

Implementierung: Frontend



Implementierung: Frontend

Ausschnitt: Shapes



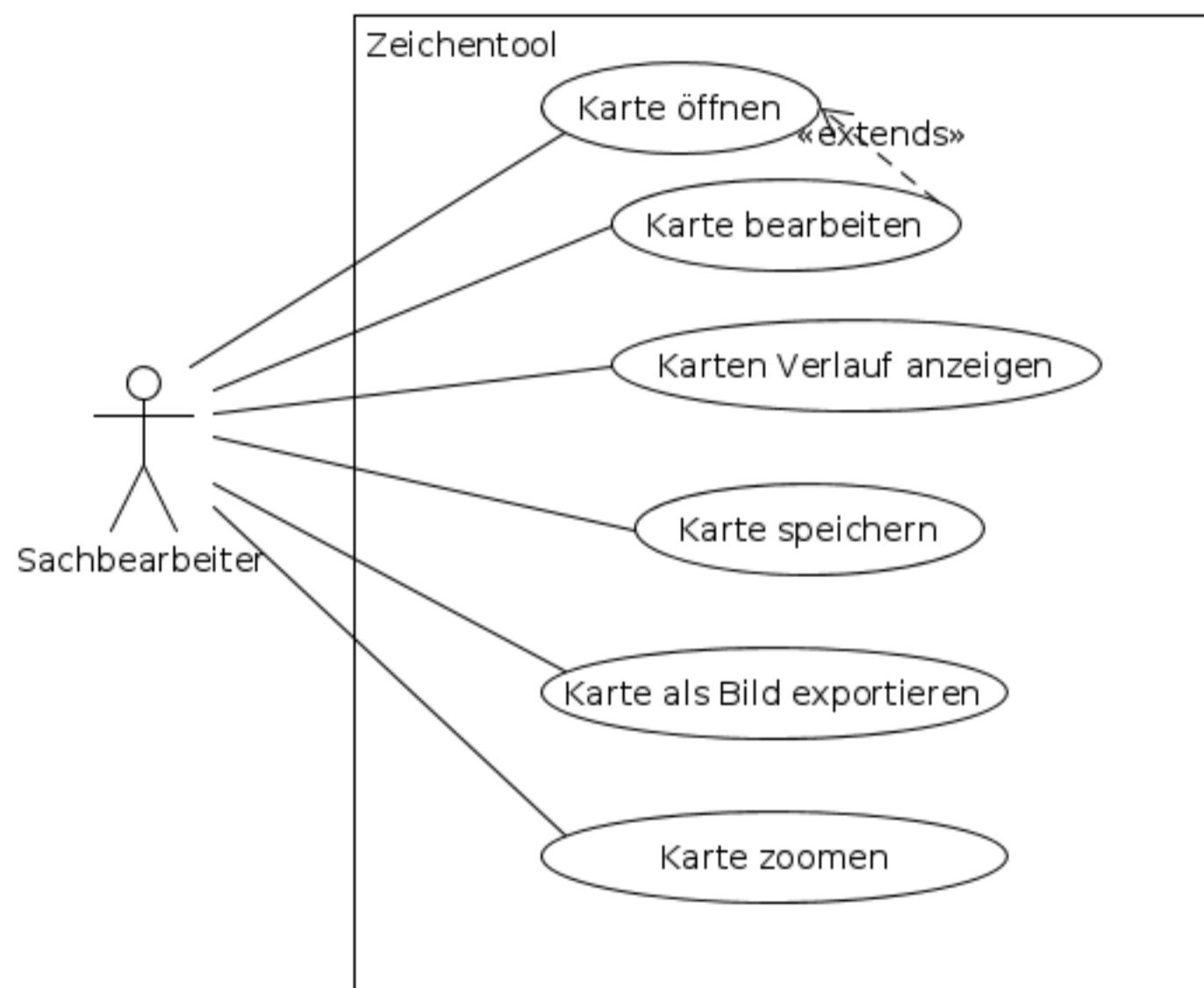
Implementierung: Frontend

Usecase Diagramme

- Grundfunktionen
- Karte bearbeiten
- Zeichenobjekte bearbeiten

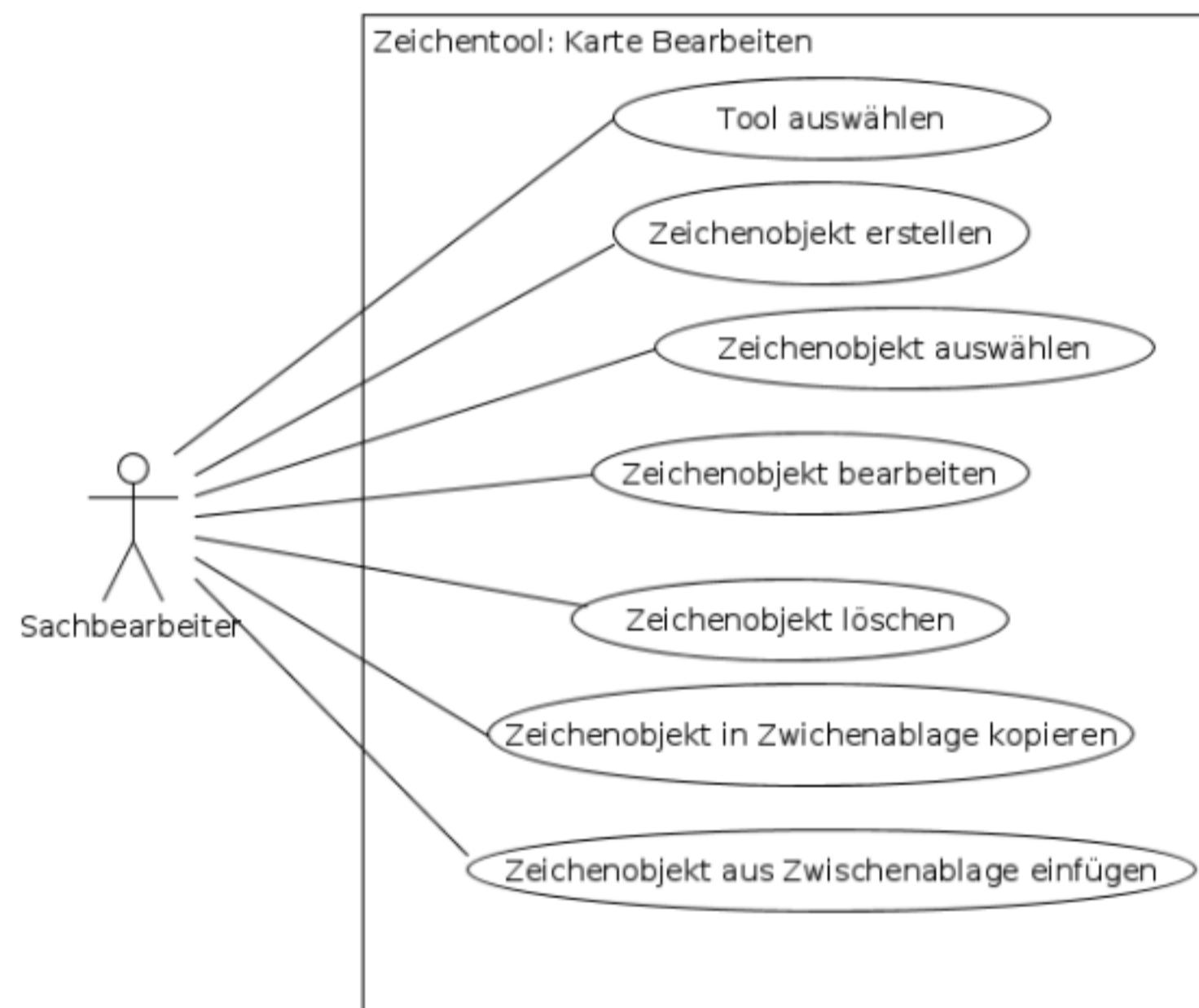
Implementierung: Frontend

Zeichentool Grundfunktionen



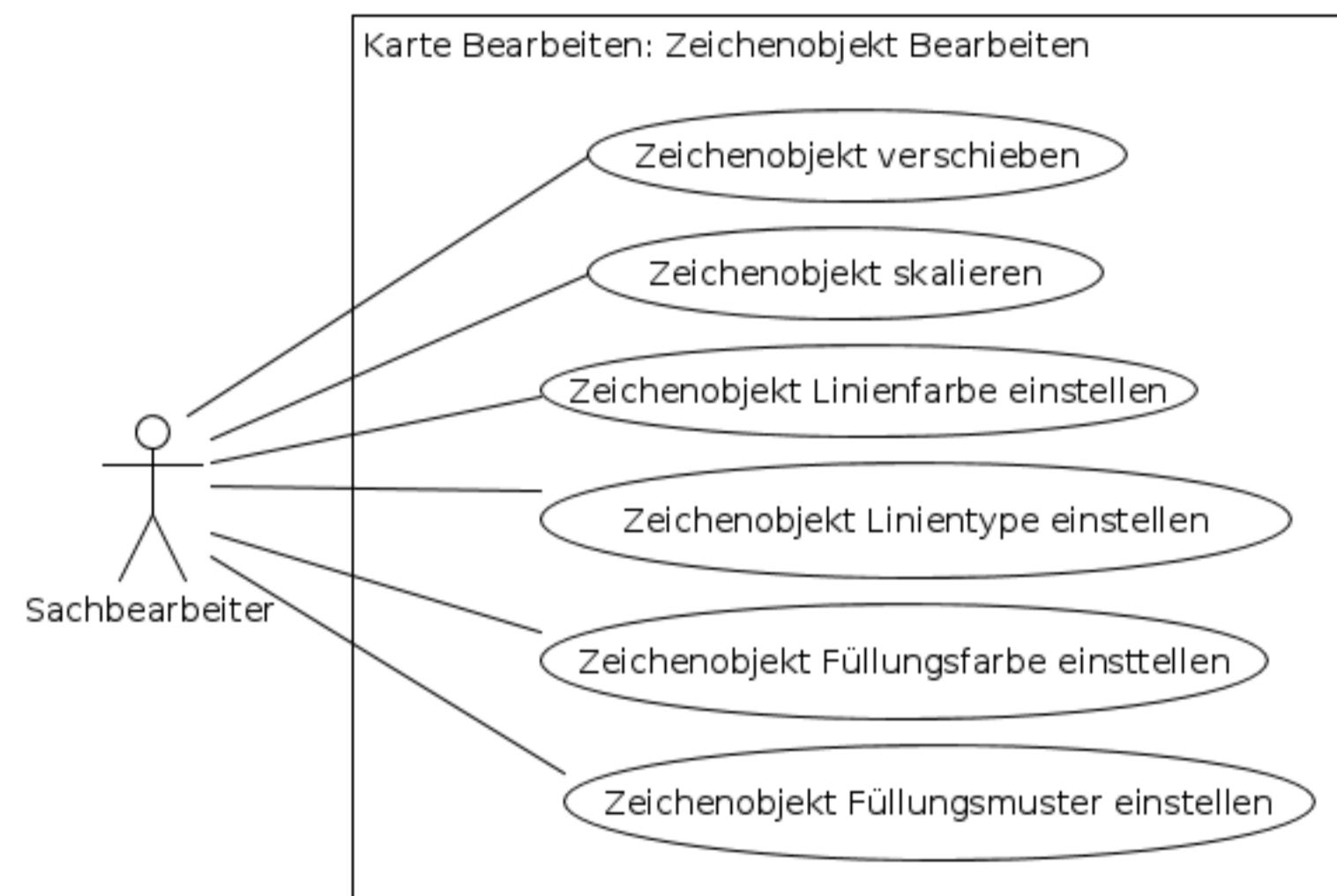
Implementierung: Frontend

Karte Bearbeiten



Implementierung: Frontend

Zeichenobjekte Bearbeiten



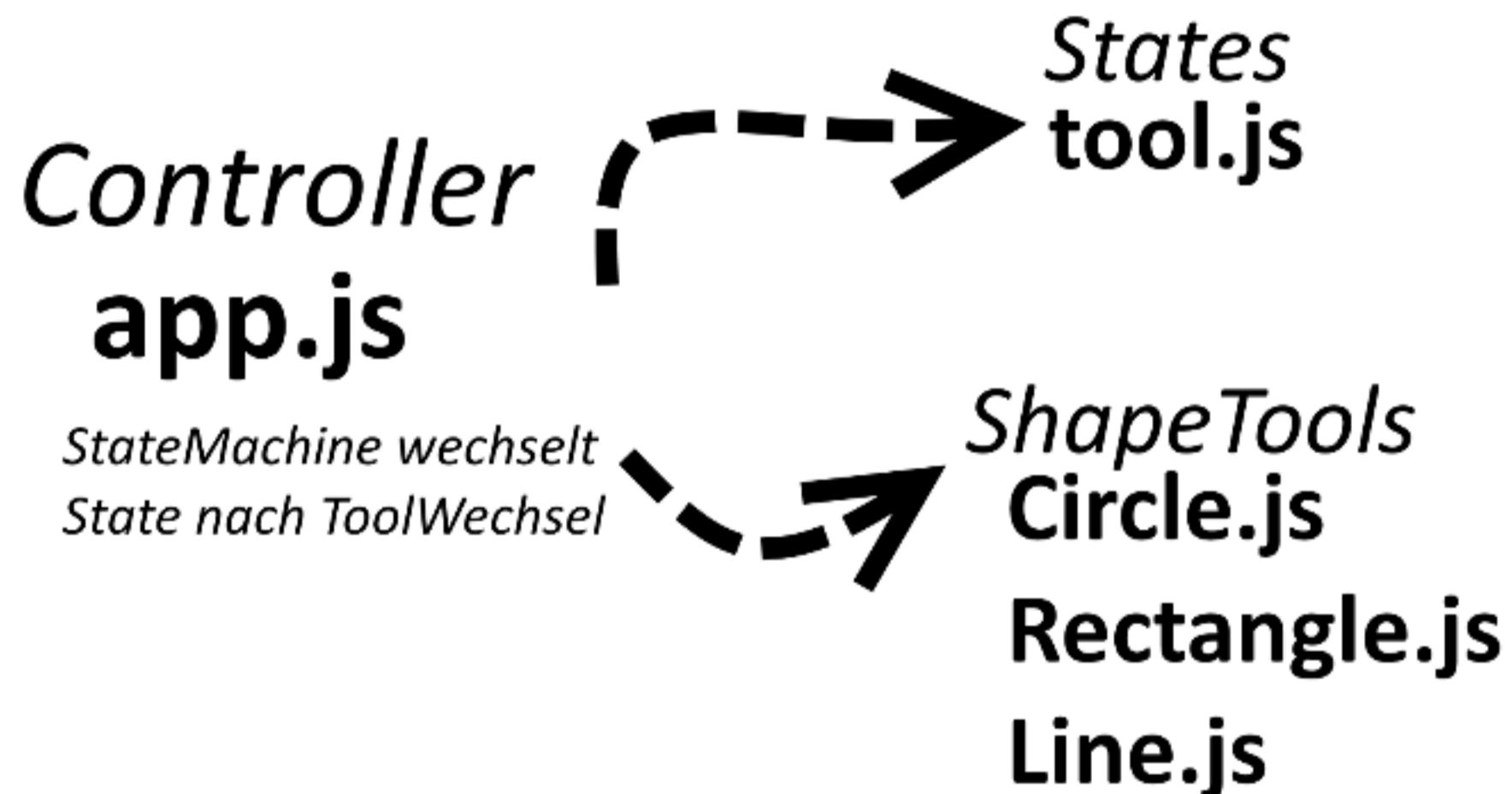
Implementierung: Frontend

Codestruktur

- Version 1
- Versionen im Überblick
- Version 3

Implementierung: Frontend

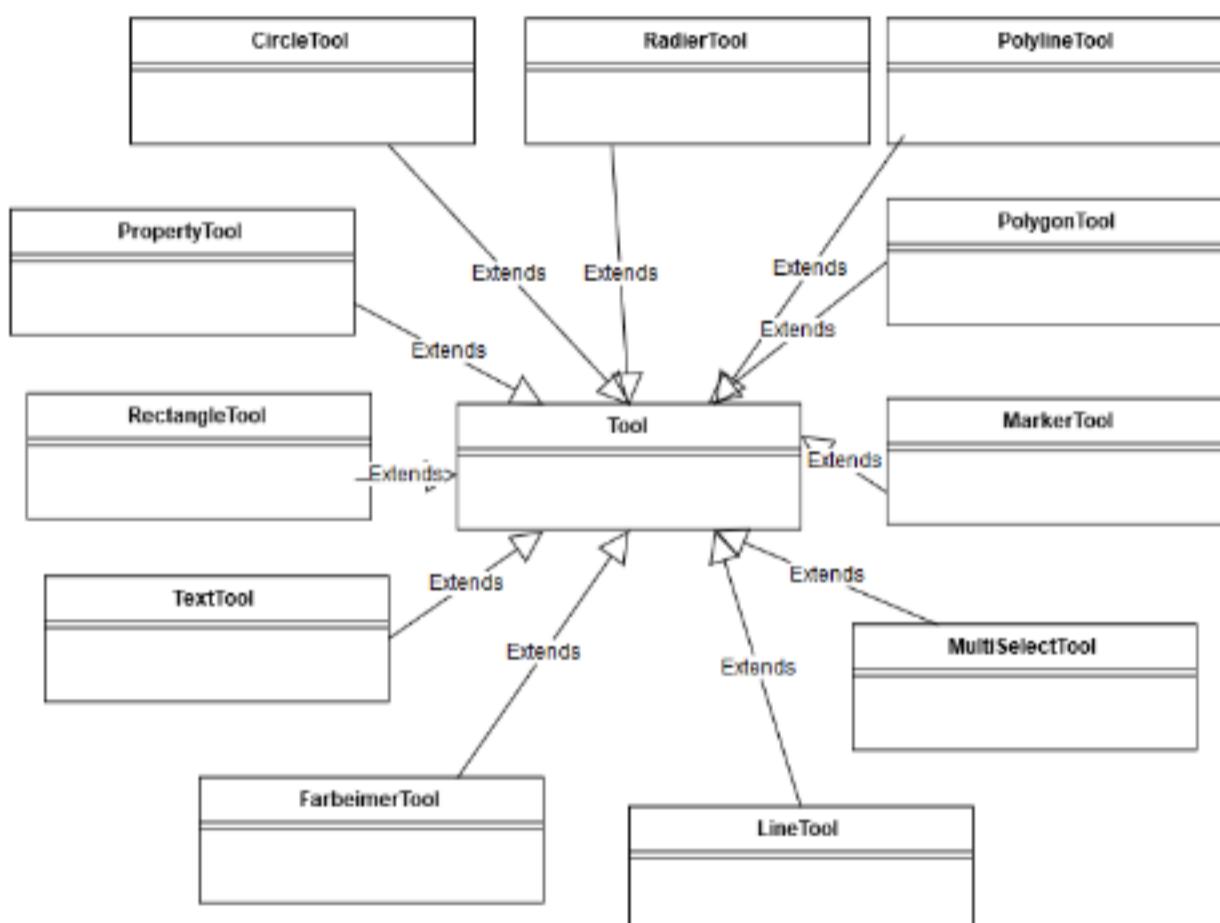
Code Version 1



Implementierung: Frontend

Klassendiagramm

Alte Variante
Keine echten Klassen
Keine Vererbung
Keine Polymorphie



Code Version 1

CodeAusschnitte

- tool.js (*States*)
- app.js (*Controller*)
- rectangle.js (*ShapeTool*)
- polygon.js (*ShapeTool*)

tool.js

```
let tool = {  
    CURSOR: 0,  
    CIRCLE: 1,  
    RECTANGLE: 2,  
    LINE: 3,  
    POLYLINE: 4,  
    POLYGON: 5,  
    MARKER: 6,  
    COLORCAN: 7  
};  
  
module.exports = tool;
```

Implementierung: Frontend

app.js

```
import {CircleDown, drawCircle, getRadiusText,
getRadiusLine} from "./tools/circle";
import {RectangleDown, drawRectangle} from
"./tools/rectangle";
import {LineDown, drawLine} from "./tools/line";
import {PolylineClick, drawPolyline, PolylinegetInit,
PolylinegetButton, PolylineReset} from "./tools/polyline";
import {PolygonClick, drawPolygon, PolygongetInit,
PolygongetButton, PolygonReset} from "./tools/polygon";
import tool from "./tool";

// // CURRENT STATE
let active tool = tool.CURSOR;
```

Implementierung: Frontend

app.js

```
// EVENT-LISTENER: MOUSE-CLICK
function mapOnClick(coordinates) {
    switch(active_tool){
        case tool.CURSOR:
            break;
        case tool.CIRCLE:
            let circle = drawCircle(coordinates.latlng);
            if(circle){
                removeAllHolograms();
                circle.addTo(mymap);
            }
            else {
                hologram_button = CircleDown(coordinates.latlng);
                hologram_button.addTo(mymap);
            }
            break;
        case tool.RECTANGLE:
            let rectangle = drawRectangle(coordinates.latlng);
            if(rectangle){
                removeHologram();
                rectangle.addTo(mymap);
            }
            else {
                RectangleDown(coordinates.latlng);
            }
            break;
        case tool.LINE:
            let line = drawLine(coordinates.latlng);
            if(line){
                removeHologram();
                line.addTo(mymap);
            }
            else {
                LineDown(coordinates.latlng);
            }
            break;
    }
}
```

Implementierung: Frontend

app.js

```
function mapOnMapMove (coordinates){  
    if(active_tool==tool.CURSOR)  
        return;  
    removeHologram();  
    switch(active_tool){  
        case tool.CIRCLE:  
            hologram = drawCircle(coordinates.latlng, true);  
            removeHologramText();  
            removeHologramLine();  
            hologram_text = getRadiusText();  
            if(hologram_text == null) return;  
            hologram_text.addTo(mymap);  
            hologram_line = getRadiusLine(coordinates.latlng);  
            hologram_line.addTo(mymap);  
            break;  
        case tool.RECTANGLE:  
            hologram = drawRectangle(coordinates.latlng, true);  
            break;  
        case tool.LINE:  
            hologram = drawLine(coordinates.latlng, true);  
            break;  
        case tool.POLYLINE:  
            hologram = drawPolyline(coordinates.latlng, true);  
            break;  
        case tool.POLYGON:  
            hologram = drawPolygon(coordinates.latlng, true);  
            break;  
    }  
    if(hologram) {  
        hologram.addTo(mymap);  
    }  
}
```

rectangle.js

```
import "leaflet";

let coordinates_first;
let drawing = false;

export function drawRectangle(coordinates_in, hologram = false){
    if(drawing==false){
        return null;
    }
    let coordinates = [coordinates_first,coordinates_in];
    let rectangle;
    if(hologram){
        rectangle = new L.rectangle(coordinates,{opacity: 0.3});
    }
    else{
        rectangle = new L.rectangle(coordinates);
        drawing = false;
    }

    return rectangle;
}

export function RectangleDown(coordinates){
    if(drawing==false) {
        coordinates_first = coordinates;
        drawing = true;
    }
}
```

Implementierung: Frontend

polygon.js

```
import "leaflet";
import {finishFormIcon} from "./icons";

let coordinates = [];
let init = true;

export function drawPolygon(holo_coor = 0,hologram = true){
    var polygon;

    if(hologram){
        if(holo_coor!=0){
            coordinates.push(holo_coor);
            polygon = new L.polyline(coordinates, {opacity: 0.3});
            coordinates.pop();
        }
        else {
            polygon = new L.polyline(coordinates, {opacity: 0.3});
        }
    }
    else{
        polygon = new L.polygon(coordinates);
        PolygonReset();
    }
    return polygon;
}

export function PolygonClick(coordinates_in){
    coordinates.push(coordinates_in);
    if(coordinates.length > 1){
        init = false;
    }
}

export function PolygongetInit(){}
```

Fazit

- doppelter Code
- unübersichtlich
- schwer erweiterbar

Ziel: OOP mit echten Klassen und Polymorphie

3 große Versionen

Version 1

- Initiales Design mit StateMachine ohne Klassen - Rückgabe von Leaflet Objects
- StateMachine mit Funktionsaufrufen aus ausgelagerten JS-Files (je ShapeTool)

Implementierung: Frontend

Version 2

- Design mit Klassen je Shape und Vererbung (ES6) - Rückgabe von Leaflet Objects
- Keine StateMachine
- Polymorphie
- Vererbungshierarchie unter den Shape-Tools

Version 2

Hierarchie

Top-Level

- Shape-Tool

Mid-Level

- MultiPoint-Shape-Tool
- TwoPoint-Shape-Tool

Low-Level

- Circle-Tool
- Rectangle-Tool
- Line-Tool
- ...

Implementierung: Frontend

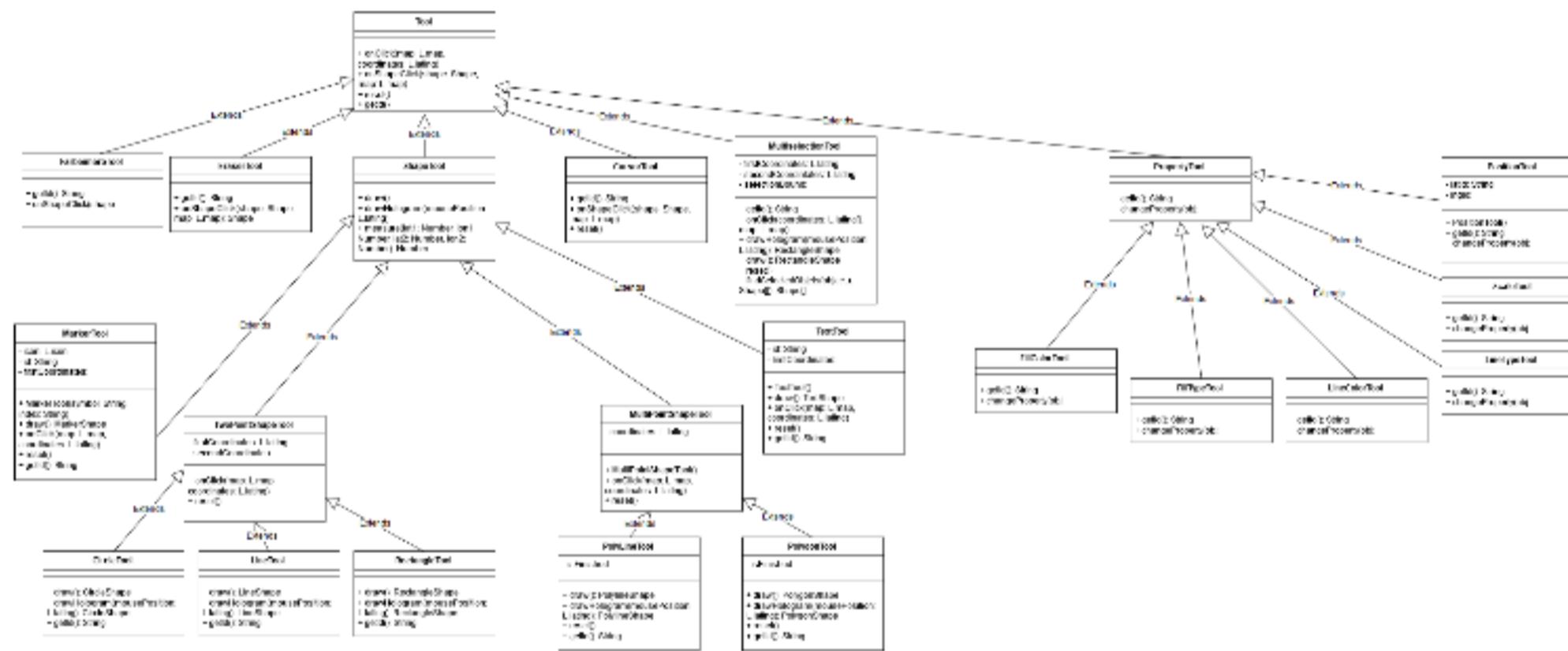
Version 3

- Rückgabe von Shapes (Eigene-Definition)
- Unterteilung in 2 Klassenhierarchien Tool & Shape
- TOOL: Erzeugung von Shapes mit ihren Koordinaten
- SHAPE: Halten des Leaflet-Objects+Styling Methoden
- => Trennung in [Circle/Rectangle...]-Tool und [Circle/Rectangle...]-Shape

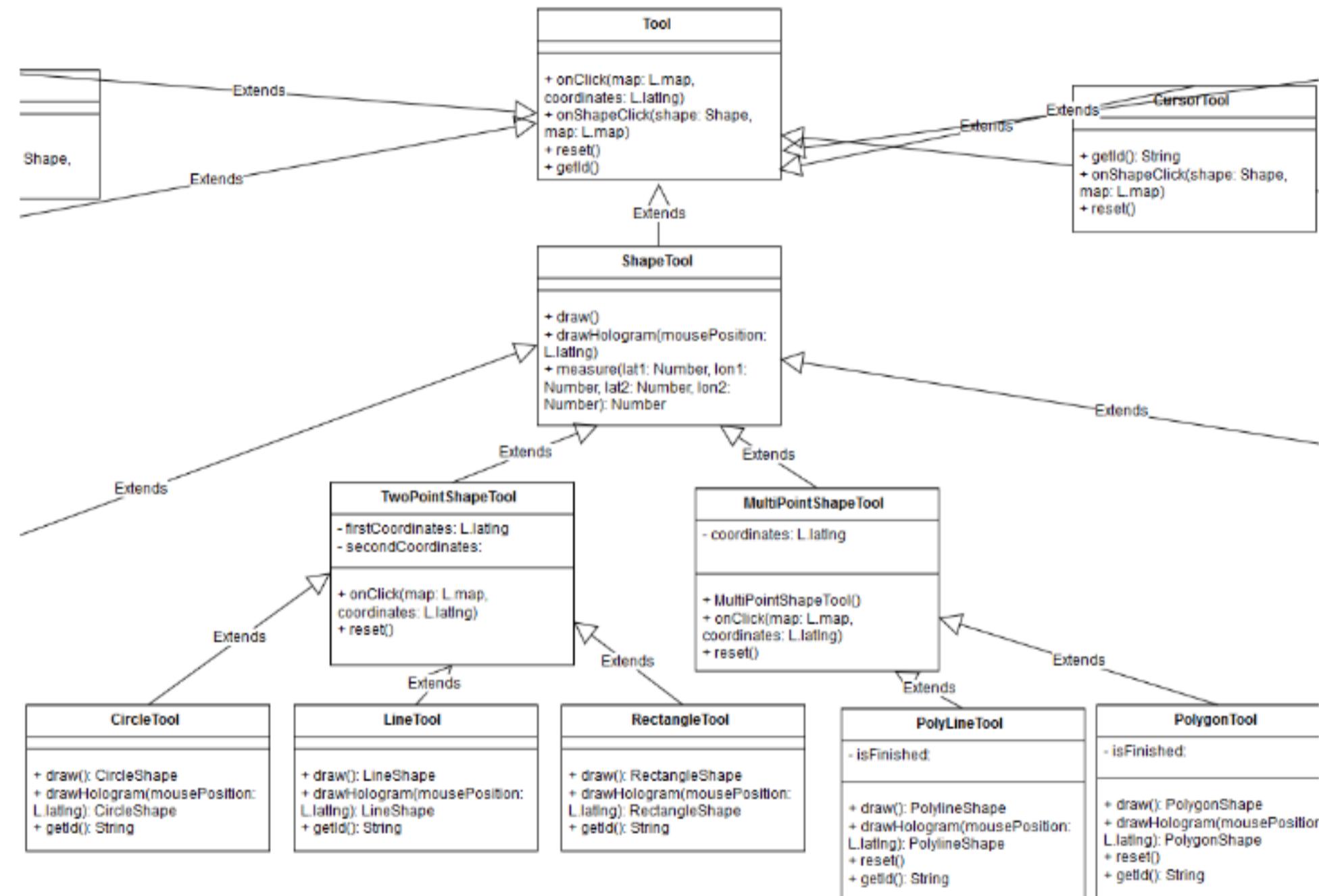
Implementierung: Frontend

Neues Klassendiagramm

Tools

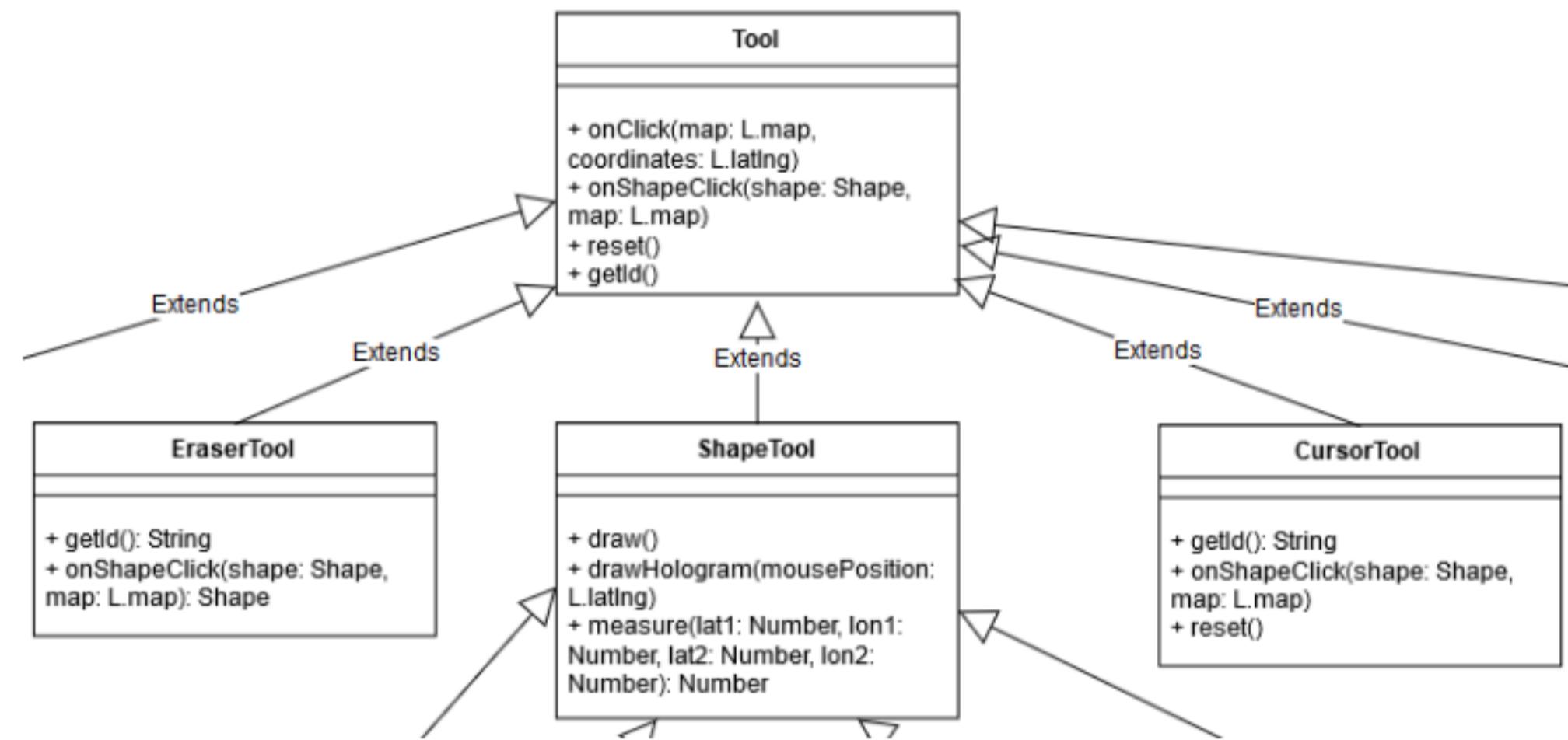


Implementierung: Frontend



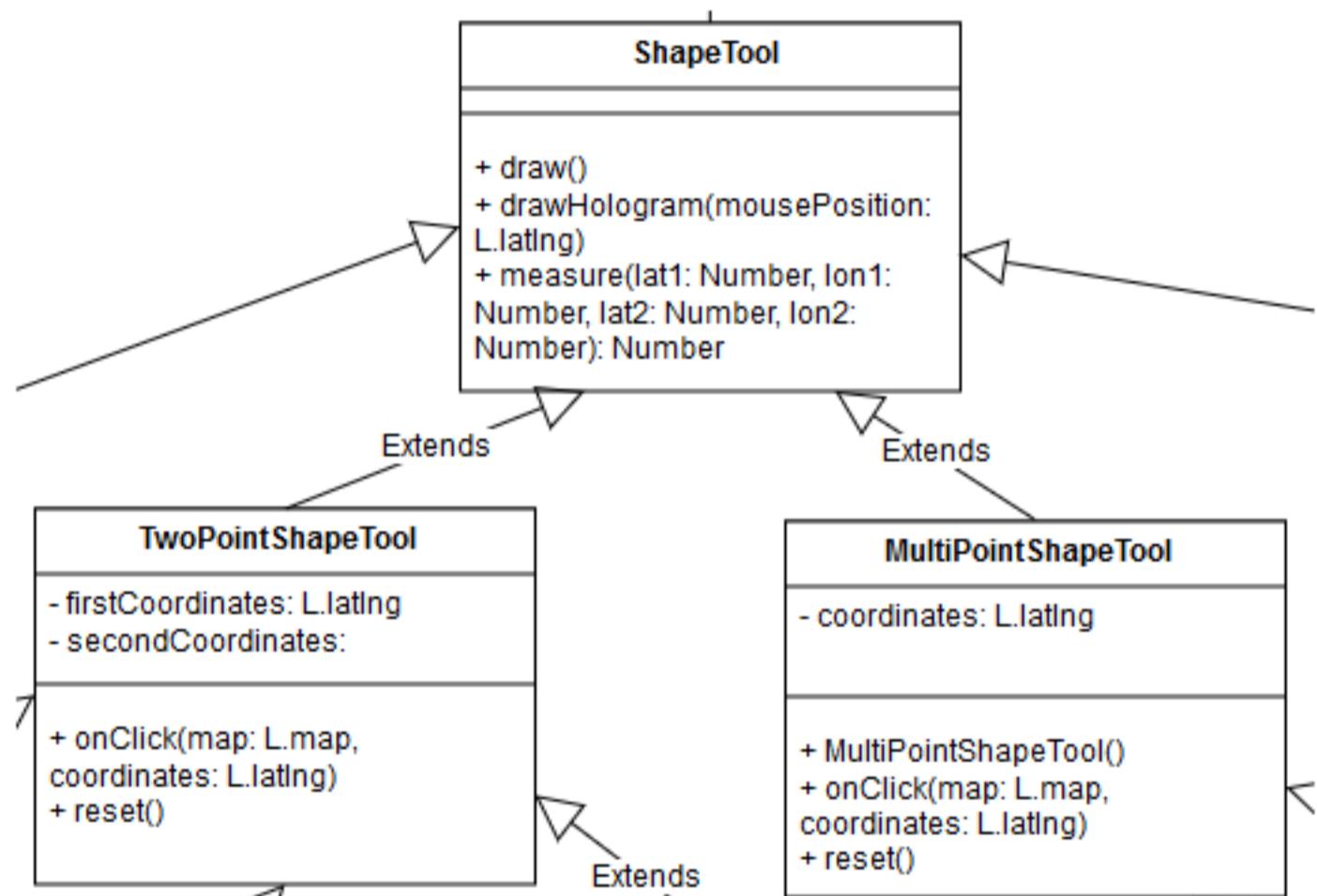
Implementierung: Frontend

Tools - Tools bereitstellen



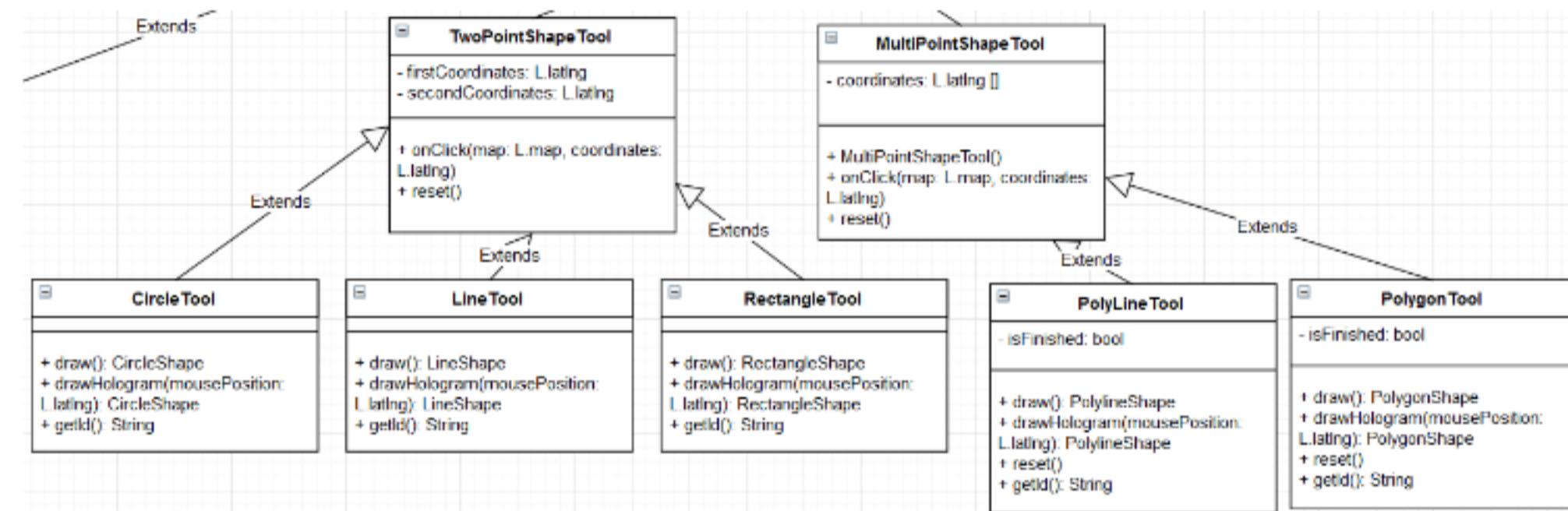
Implementierung: Frontend

PointTools - Shapes einteilen



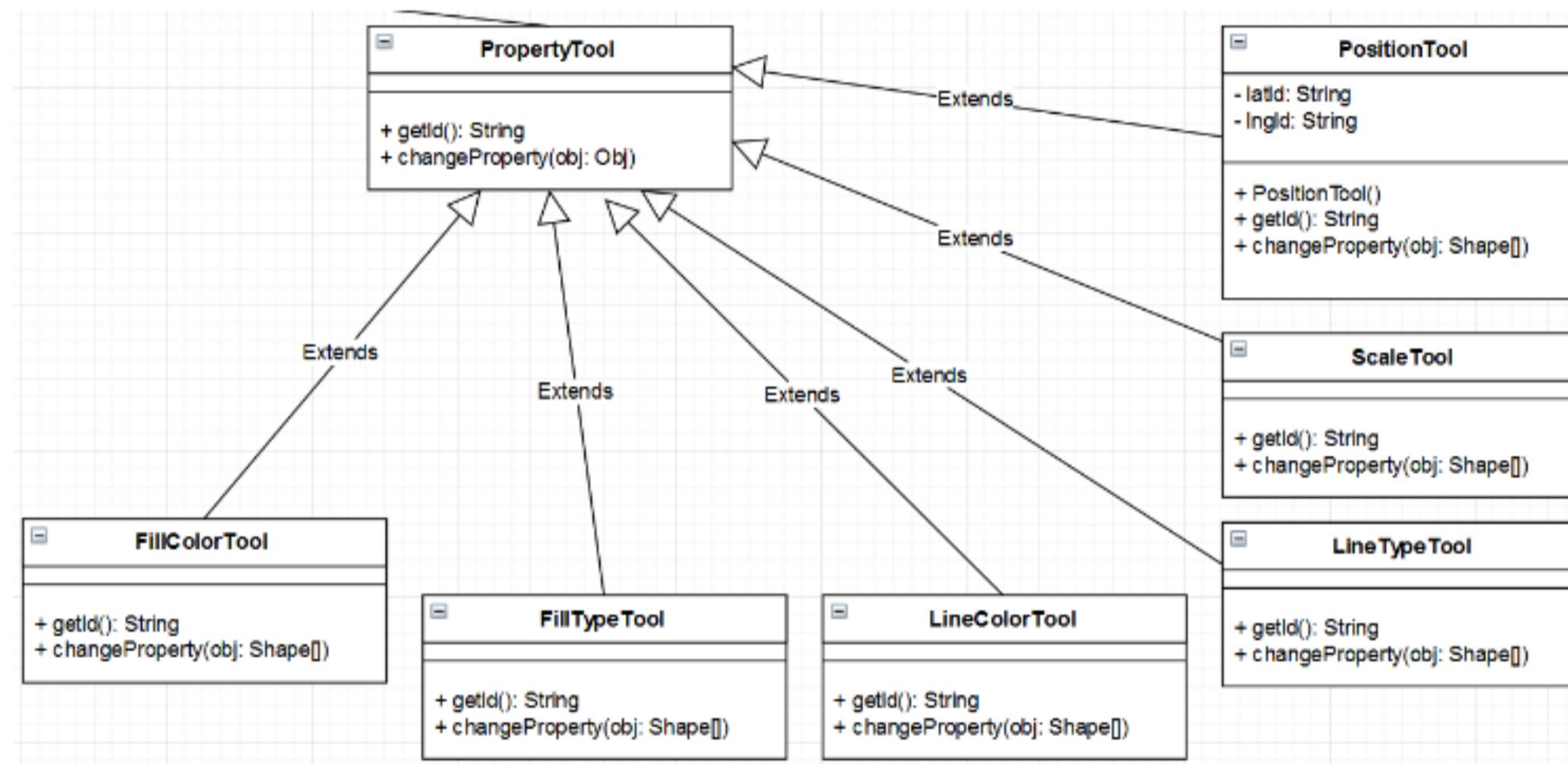
Implementierung: Frontend

ShapeTools - Shapes erzeugen

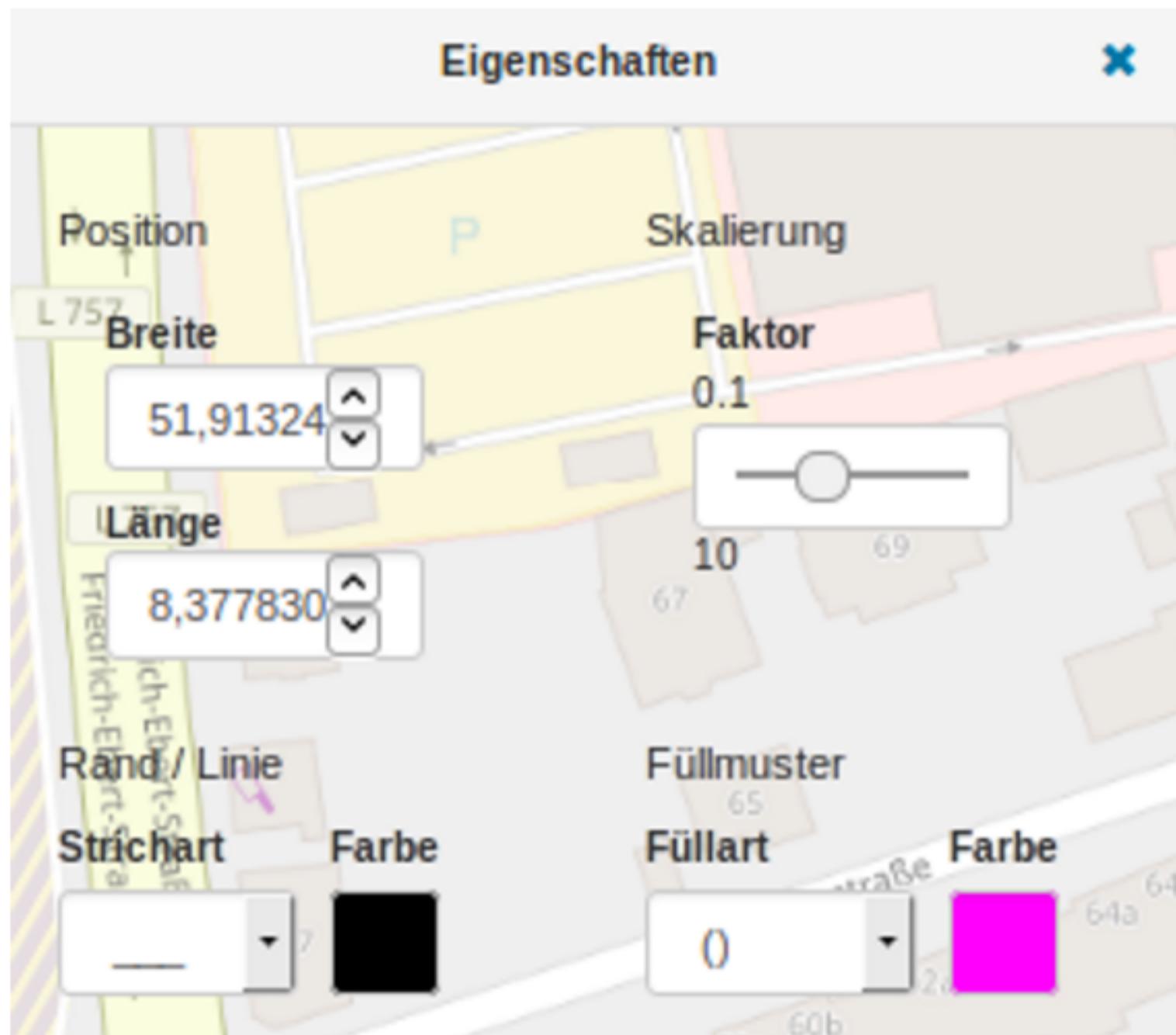


Implementierung: Frontend

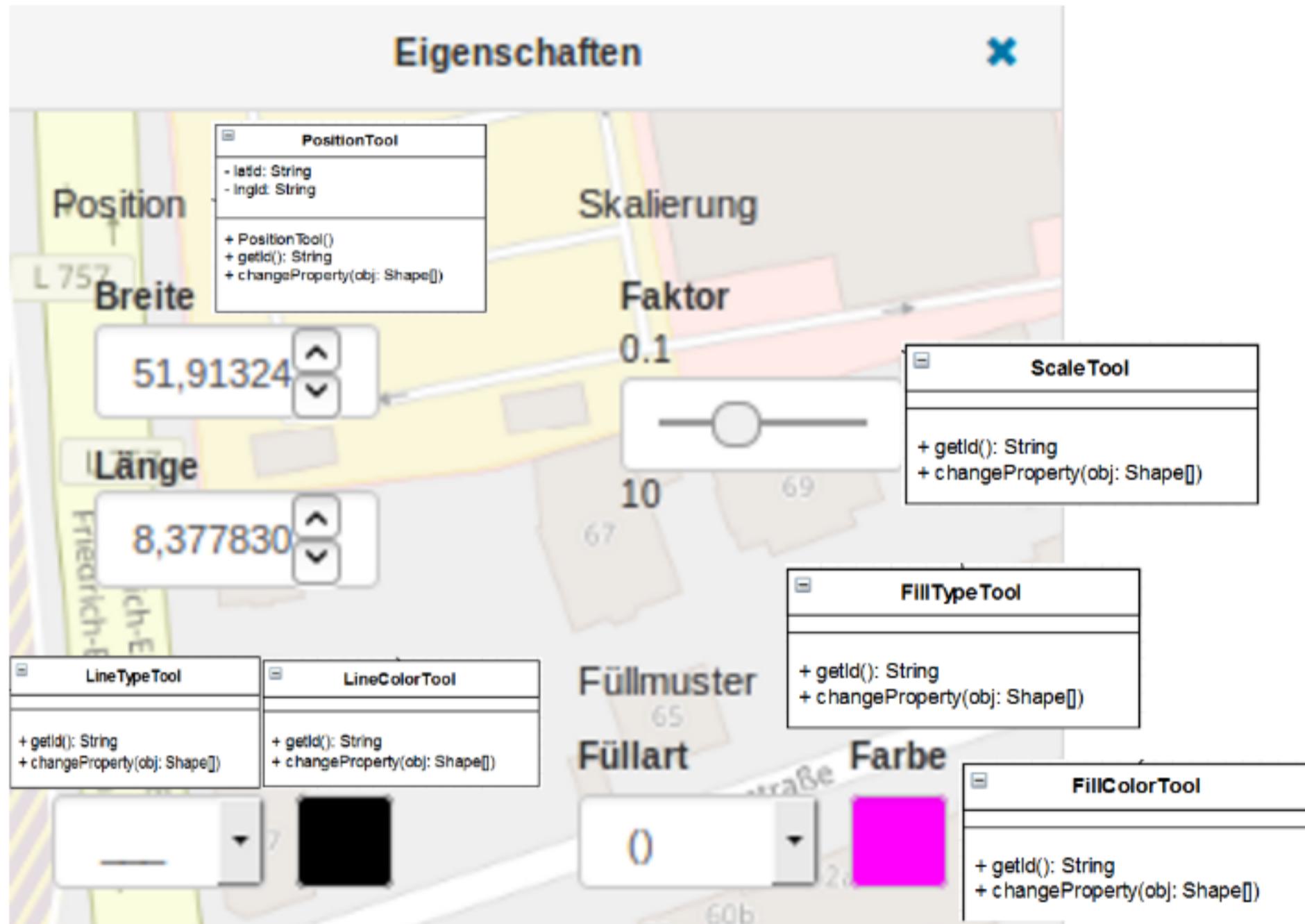
PropertyTools - zum Styling



Implementierung: Frontend

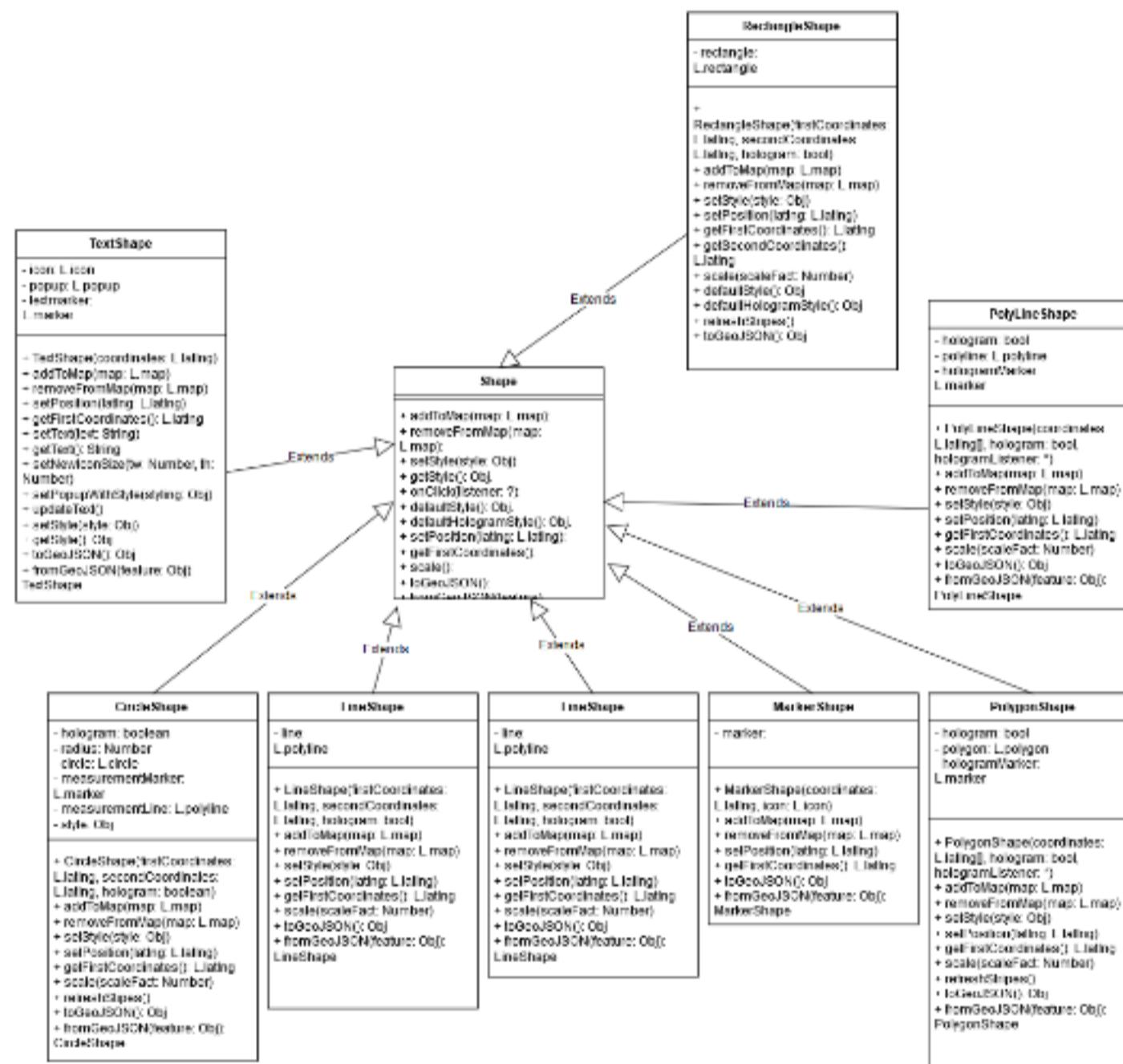


Implementierung: Frontend



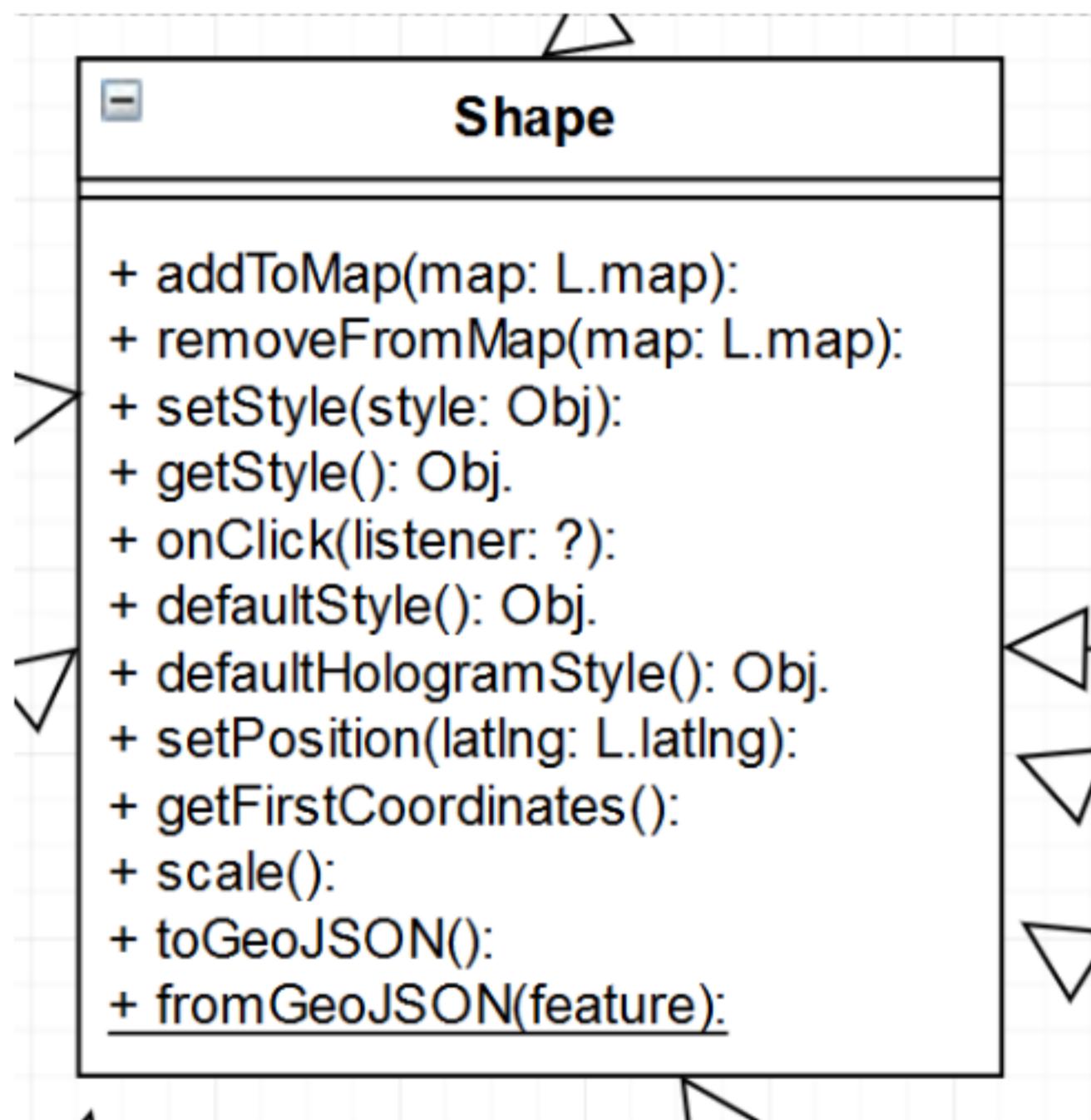
Implementierung: Frontend

Shapes



Implementierung: Frontend

Shape



Implementierung: Frontend

Circle und Line Shapes

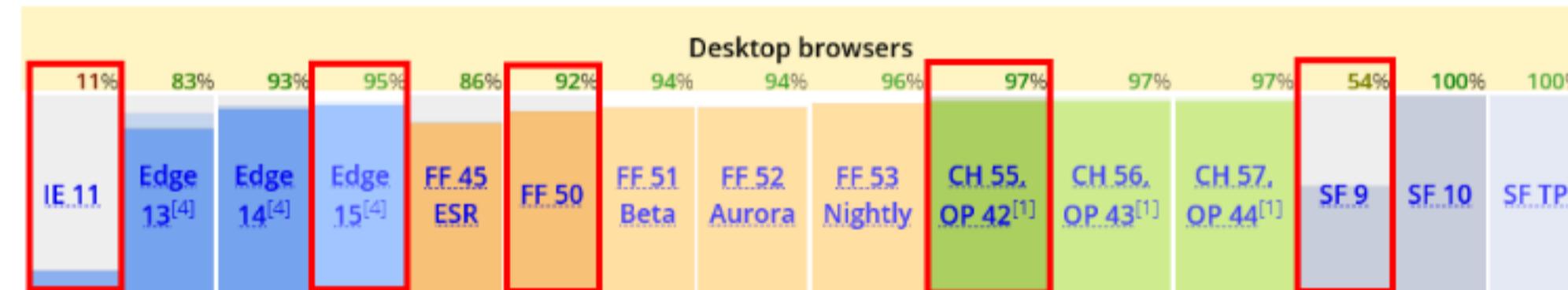
CircleShape	LineShape
<ul style="list-style-type: none">- hologram: boolean- radius: Number- circle: L.circle- measurementMarker: L.marker- measurementLine: L.polyline- style: Obj	<ul style="list-style-type: none">- line: L.polyline- style: Obj
<ul style="list-style-type: none">+ CircleShape(firstCoordinates: L.latlng, secondCoordinates: L.latlng, hologram: boolean)+ addToMap(map: L.map)+ removeFromMap(map: L.map)+ setStyle(style: Obj)+ setPosition(latlng: L.latlng)+ getFirstCoordinates(): L.latlng+ scale(scaleFact: Number)+ refreshStipes()+ toGeoJSON(): Obj+ fromGeoJSON(feature: Obj): CircleShape	<ul style="list-style-type: none">+ LineShape(firstCoordinates: L.latlng, secondCoordinates: L.latlng, hologram: bool)+ addToMap(map: L.map)+ removeFromMap(map: L.map)+ setPosition(latlng: L.latlng)+ getFirstCoordinates(): L.latlng+ scale(scaleFact: Number)+ toGeoJSON(): Obj+ fromGeoJSON(feature: Obj): LineShape

Implementierung: Frontend

ES6

ECMA Script 6

- liefert Standards für u.a. JavaScript
- Finaler Release: Jun.2015
- Neueste Version: 7 - ECMAScript 2016 (Jun.2016)



Implementierung: Frontend

ECMA Script 6 Feature Auswahl

- Konstanten

```
const PI = 3.141593
PI > 3.0
```

- Scopes (Variablen/Funktionen)

```
for (let i = 0; i < a.length; i++) {
    let x = a[i]
```

- Defaultparameter

```
function f (x, y = 7, z = 42) {
    return x + y + z
}
```

Implementierung: Frontend

- Exports/Imports

```
// lib/math.js
export function sum (x, y) { return x + y }
export var pi = 3.141593

// someApp.js
import * as math from "lib/math"
console.log("2π = " + math.sum(math.pi, math.pi))

// otherApp.js
import { sum, pi } from "lib/math"
console.log("2π = " + sum(pi, pi))
```

- Arrow Functions

```
nums.forEach(v => {
  if (v % 5 === 0)
    fives.push(v)
})
```

Implementierung: Frontend

Klassen statt Prototypes

- Klasse definieren

```
class Shape {  
    constructor (id, x, y) {  
        this.id = id  
        this.move(x, y)  
    }  
    move (x, y) {  
        this.x = x  
        this.y = y  
    }  
}
```

- Vererbung

```
class Rectangle extends Shape {  
    constructor (id, x, y, width, height) {  
        super(id, x, y)  
        this.width = width  
        this.height = height  
    }  
}
```

- Get/Set innerhalb der Klasse

Implementierung: Frontend

Neue code Struktur

Tool

```
export default class Tool {  
  
    onClick(map, coordinates) {  
    }  
  
    onShapeClick(shapes, map) {  
    }  
  
    reset() {  
    }  
  
    getId() {  
    }  
  
    showHint(text){  
        Hint.showHint(text);  
    }  
};
```

Implementierung: Frontend

Tool

ShapeTool

```
export default class ShapeTool extends Tool {

    draw() {
    }

    drawHologram(mousePosition) {
    }

    /**
     * Static helper method for measuring.
     */
    static measure(lat1, lon1, lat2, lon2) {
        const R = 6378.137; // Radius of earth in KM
        const dLat = lat2 * Math.PI / 180 - lat1 * Math.PI / 180;
        const dLon = lon2 * Math.PI / 180 - lon1 * Math.PI / 180;
        const a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
            Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
            Math.sin(dLon / 2) * Math.sin(dLon / 2);
        const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
        const d = R * c;

        return d * 1000; // meters
    }
}
```

Implementierung: Frontend

Tool

TwoPointShapeTool

```
export default class TwoPointShapeTool extends ShapeTool {

    onClick(map, coordinates) {
        if (!this.firstCoordinates) {
            this.firstCoordinates = coordinates;
        } else if (!this.secondCoordinates) {
            this.secondCoordinates = coordinates;
        }
        this.showHint();
    }

    reset() {
        this.firstCoordinates = null;
        this.secondCoordinates = null;
    }
}
```

Implementierung: Frontend

Tool

RectangleShapeTool

```
export default class RectangleTool extends TwoPointShapeTool {

    draw() {
        if (this.firstCoordinates && this.secondCoordinates) {
            return new RectangleShape(this.firstCoordinates, this.secondCoordinates);
        }
    }

    drawHologram(mousePosition) {
        if (this.firstCoordinates) {
            return new RectangleShape(this.firstCoordinates, mousePosition, true);
        }
    }

    showHint() {
        if(!this.firstCoordinates){
            super.showHint("Klicken Sie auf die Karte um den Anfangspunkt des Rechtecks auszuwählen.");
        } else if(!this.secondCoordinates){
            super.showHint("Ziehen sie mit der Maus um das entsprechend Rechteck zu zeichnen. " +
                "Klicken sie erneut um das Rechteck abzuschließen.");
        } else {
            super.showHint("Sie können weitere Rechtecke zeichnen. " +
                "Um das Aussehen des Rechtecks zu verändern, wählen Sie ein Werkzeug aus.");
        }
    }

    getId() {
        return "#toolRectangle";
    }
}
```

Implementierung: Frontend

Shape

Shape

```
export default class Shape {

    addToMap(map) {
    }

    removeFromMap(map) {
    }

    setStyle(style) {
    }

    getStyle() {
        return this.style;
    }

    onClick(listener) {
        this.listener = listener;
    }

    /**
     * Returns the default style for this shape.
     * @returns {{color: (*|jQuery)}} The default style.
     */
    defaultStyle() {
    }
}
```

Implementierung: Frontend

Shape

RectangleShape

```
export default class RectangleShape extends Shape {

    constructor(firstCoordinates, secondCoordinates, hologram = false) {
        super();

        this.rectangle = L.rectangle([firstCoordinates, secondCoordinates]);

        if (hologram) {
            this.setStyle(this.defaultHologramStyle());
        } else {
            this.rectangle.on("click", () => {
                this.listener();
            });

            this.setStyle(this.defaultStyle());
        }
    }

    addToMap(map) {
        this.map = map;

        this.rectangle.addTo(map);

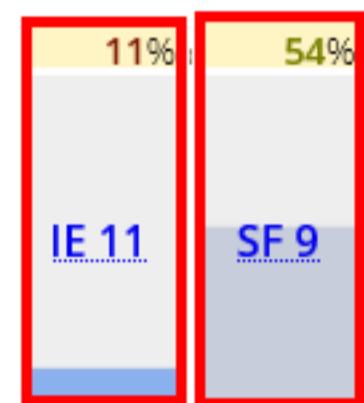
        this.refreshStripes();
    }

    removeFromMap(map) {
        this.rectangle.remove(map);
    }
}
```

BABEL

- Aktuelle Version: 6.19.0

Motivation



- Alte Browser unterstützen, neue Features beibehalten
- Transpiliert ES6 Standard in niedrigeren ES5

Implementierung: Frontend

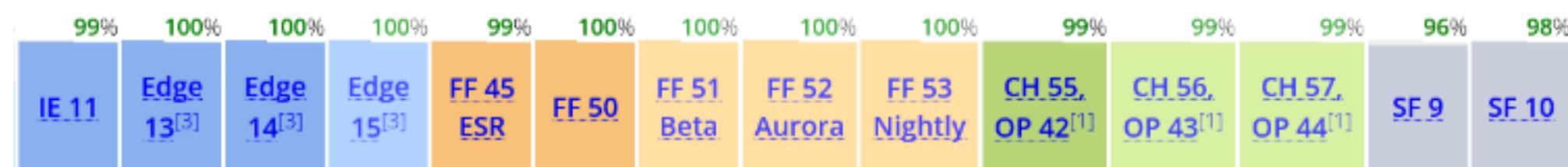
ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
function f (x, y = 7, z = 42) {  
    return x + y + z  
}  
f(1) === 50
```

ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
function f (x, y, z) {  
    if (y === undefined)  
        y = 7;  
    if (z === undefined)  
        z = 42;  
    return x + y + z;  
};  
f(1) === 50;
```

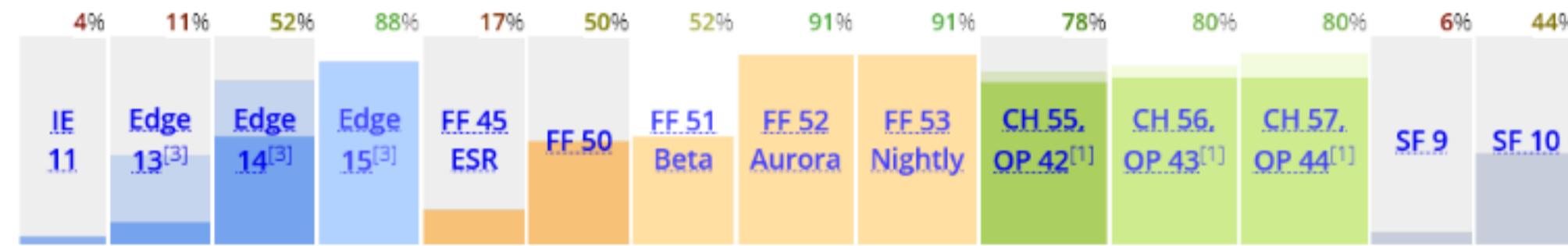
ES5 Support



Implementierung: Frontend

ES7 existiert bereits und ES8 ist in Planung

ES7 Support



Implementierung: Frontend

GeoJSON

Das GeoJSON Format wurde für die verschiedenen Shapes erweitert:

GeoJSON Objekt	Shape
Circle	CircleShape
Rectangle	RectangleShape
Marker	MarkerShape
Text	TextShape

Implementierung: Frontend

Konvertieren zu GeoJSON: Circle

```
toGeoJSON() {
  const feature = {
    type: "Feature",
    geometry: {
      type: "",
      coordinates: []
    },
    properties: {}
  };

  feature.properties.radius = this.circle.getRadius();
  feature.properties.style = this.getStyle();

  if (feature.properties.style && feature.properties.style.fillPattern) {
    feature.properties.style.fillPattern = {
      options: feature.properties.style.fillPattern.options
    };
  }

  feature.geometry.type = "Circle";
  feature.geometry.coordinates = [
    this.getFirstCoordinates().lng,
    this.getFirstCoordinates().lat
  ];

  return feature;
}
```

Implementierung: Frontend

Konvertieren von GeoJSON: Circle

```
static fromGeoJSON(feature) {
    const coordinates = {
        lng: feature.geometry.coordinates[0],
        lat: feature.geometry.coordinates[1]
    };

    const shape = new CircleShape(coordinates, coordinates);
    shape.circle.setRadius(feature.properties.radius);

    if (feature.properties.style) {
        if (feature.properties.style.fillPattern && feature.properties.style.fillPattern.options) {
            feature.properties.style.fillPattern = new L.StripePattern(feature.properties.style.fillPattern.options);
        }
        shape.setStyle(feature.properties.style);
    }

    return shape;
}
```

Backend

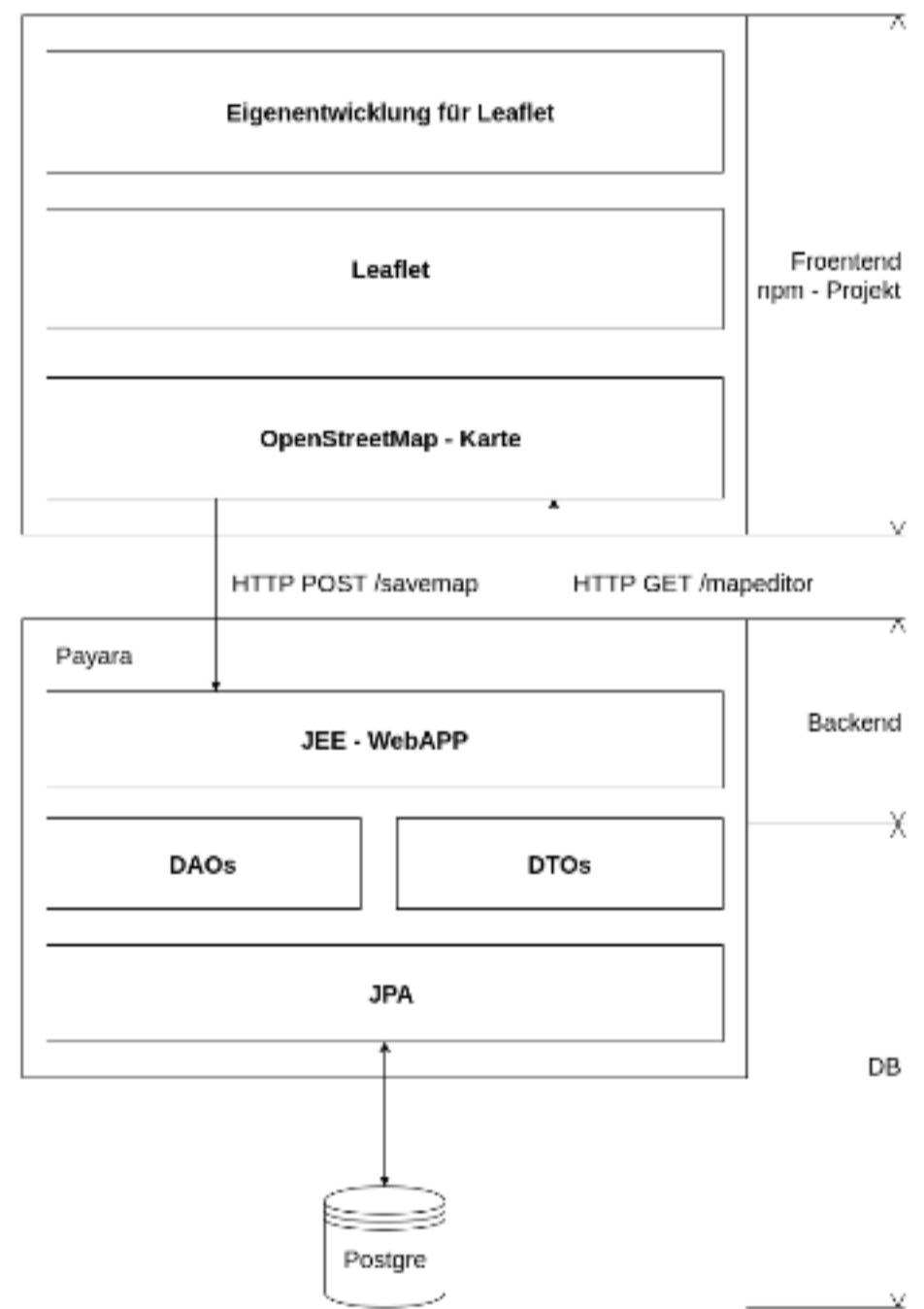
Implementierung: Backend

Technologie Überblick

- Servlets
- JSF
- JPA
- Apache Commons IO
- Apache Commons Lang
- Gson
- Gradle

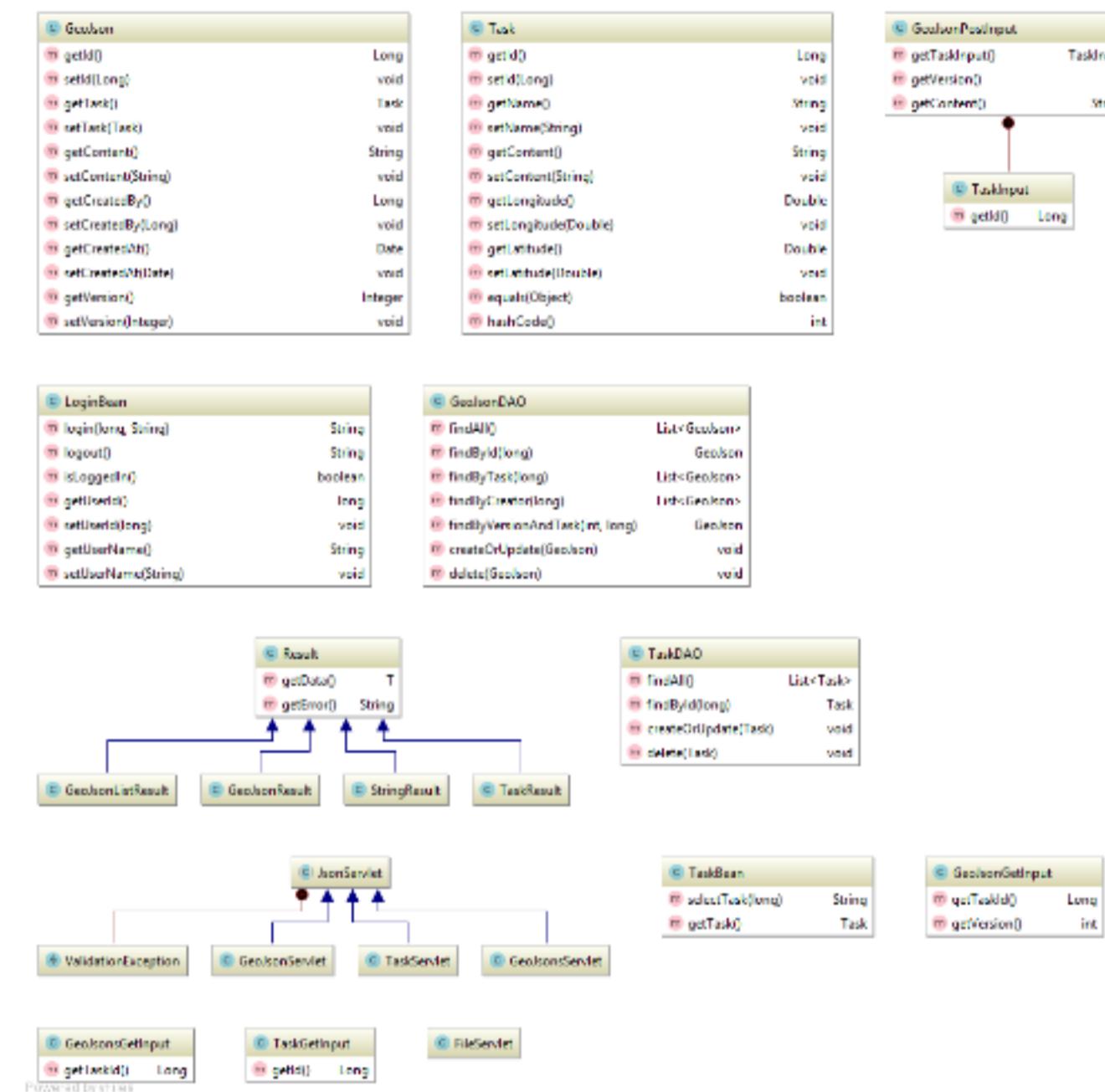
Implementierung: Backend

Architektur



Implementierung: Backend

Klassendiagramm



Powered by yEd

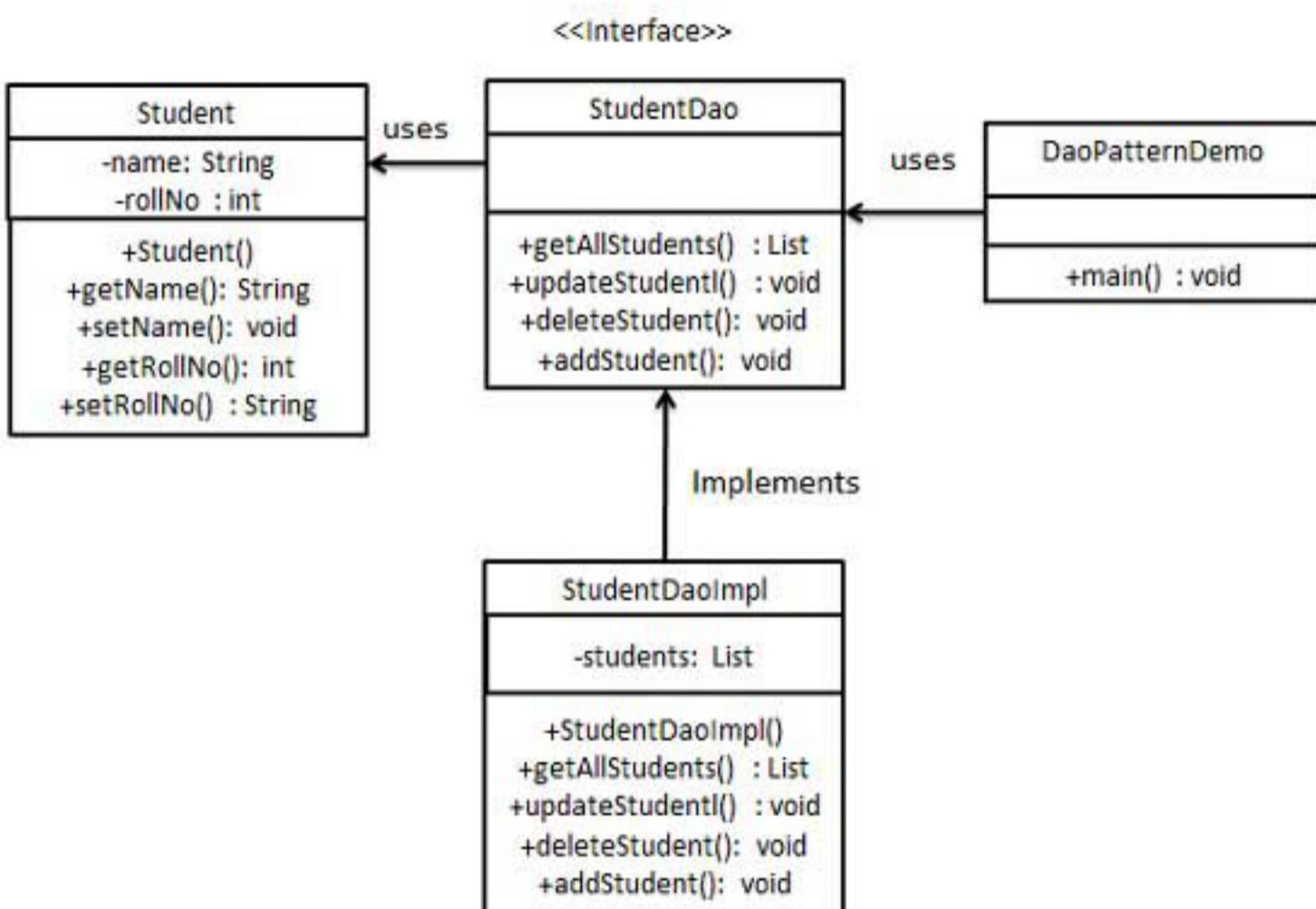
Implementierung: Backend

Datenbank-Schema

TASK			GEOJSON		
PK	id	INT	PK	id	INT
NOT_NULL	name	STRING	FK, UNIQUE(task_id, version)	task_id	INT
	content	STRING	NOT_NULL	content	STRING
NOT_NULL	longitude	DECIMAL	FK	created_by	INT
NOT_NULL	latitude	DECIMAL	NOT_NULL	created_at	DATETIME
			NOT_NULL, UNIQUE(task_id, version)	version	INT

Implementierung: Backend

DAO Pattern



Implementierung: Backend

Beispiel für eine Entity: GeoJson

```
/*
@Entity
@Table(name = "geojson", uniqueConstraints = @UniqueConstraint(columnNames = {"task_id", "version"}))
@NamedQueries({
    @NamedQuery(name = "GeoJson.findAll", query = "SELECT gj FROM GeoJson gj"),
    @NamedQuery(name = "GeoJson.findById", query = "SELECT gj FROM GeoJson gj WHERE gj.id = :id"),
    @NamedQuery(name = "GeoJson.findByTask", query = "SELECT gj FROM GeoJson gj WHERE gj.task.id = :id"),
    @NamedQuery(name = "GeoJson.findByCreator", query = "SELECT gj FROM GeoJson gj WHERE gj.createdBy = :id"),
    @NamedQuery(name = "GeoJson.findByVersionAndTask", query = "SELECT gj FROM GeoJson gj " +
        "WHERE gj.version = :version AND gj.task.id = :id")
})
public class GeoJson implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    @Expose
    private Long id;

    @ManyToOne(optional = false)
    @JoinColumn(name = "task_id", nullable = false)
    @Expose
    private Task task;

    @Column(name = "content", nullable = false, length = Short.MAX_VALUE)
    @Expose
    private String content;

    @Column(name = "created_by", nullable = false)
    @Expose
    private Long createdBy;

    @Column(name = "created_at", nullable = false)
    @Temporal(TemporalType.TIMESTAMP)
    @Expose
    private Date createdAt;

    @Column(name = "version", nullable = false)
    @Expose
    private Integer version;
}
```

Implementierung: Backend

Beispiel für ein DAO: GeoJsonDAO

```
/*
 * Finds all objects in the database.
 *
 * @return The objects.
 */
@Transactional
public List<GeoJson> findAll() {
    return entityManager.createNamedQuery( name: "GeoJson.findAll" , GeoJson.class).getResultList();
}

/**
 * Finds an object by its id.
 *
 * @param id The id.
 * @return The found object or null, if none exists.
 */
@Transactional
public GeoJson findById(long id) {
    return entityManager.createNamedQuery( name: "GeoJson.findById" , GeoJson.class)
        .setParameter(PARAMETER_ID, id).getSingleResult();
}

/**
 * Finds objects by their associated task.
 *
 * @param task_id The id of the task.
 * @return The found objects.
 */
@Transactional
public List<GeoJson> findByTask(long task_id) {
    return entityManager.createNamedQuery( name: "GeoJson.findByTask" , GeoJson.class)
        .setParameter(PARAMETER_ID, task_id).getResultList();
}
```

Implementierung: Backend

Beispiel für ein DAO: GeoJsonDAO

```
/**  
 * Creates a new entry or updates it, if already exists.  
 *  
 * @param geoJson The object.  
 */  
@Transactional  
public void createOrUpdate(GeoJson geoJson) { entityManager.persist(geoJson); }  
  
/**  
 * Deletes the object.  
 *  
 * @param geoJson The object.  
 */  
@Transactional  
public void delete(GeoJson geoJson) { entityManager.remove(geoJson); }
```

Implementierung: Backend

Besonderheit @Transactional

```
public List<GeoJson> findAllNoAnnotation() {
    List<GeoJson> result;

    entityManager.getTransaction().begin();

    result = entityManager.createNamedQuery( name: "GeoJson.findAll", GeoJson.class).getResultList();
    |
    entityManager.getTransaction().commit();
    entityManager.close();

    return result;
}
```

Vs.

```
@Transactional
public List<GeoJson> findAll() {
    return entityManager.createNamedQuery( name: "GeoJson.findAll", GeoJson.class).getResultList();
}
```

Implementierung: Backend

Exkurs: Gson

Gson ist eine Bibliothek von Google, die es ermöglicht, Objekte von und zu JSON zu konvertieren.

```
public class Staff {  
  
    private String name;  
    private int age;  
    private String position;  
    private BigDecimal salary;  
    private List<String> skills;  
  
    //...  
  
    //1. Convert object to JSON string  
    Gson gson = new Gson();  
    String json = gson.toJson(staff);  
    System.out.println(json);
```

Output

```
{"name": "mkyong", "age": 35, "position": "Founder", "salary": 10000, "skills": ["java", "python", "shell"]}
```

Implementierung: Backend

Anwendung im Servlet

```
@Inject
protected GeoJsonDAO geoJsonDAO;

@Inject
private TaskDAO taskDAO;

@Inject
private LoginBean loginBean;

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    response.setCharacterEncoding(CHARSET);

    try {
        GeoJsonGetInput input = validateAndParseGet(request);

        gson.toJson(new GeoJsonResult(geoJsonDAO.findByVersionAndTask(input.getVersion(), input.getTaskId(),
            error: null), response.getWriter());
    } catch (NoResultException exception) {
        printError(response, ERROR_NO_RESULT);
    } catch (Exception exception) {
        printError(response, exception.getMessage());
    }
}
```



Build und Deployment

Build und Deployment

Überblick

- Wie baut man ein Projekt?
- Build-Tools (npm, Gradle)
- Container (Docker)
- Unser Build- und Stage-System

Build und Deployment

Projekt bauen

Wie baut man ein Projekt?

Build und Deployment

Projekt bauen

Variante 1: IDE verwenden

Projekt bauen

- IDEs
 - Eclipse
 - IntelliJ
 - WebStorm
 - NetBeans
 - Visual Studio
 - ...

Build und Deployment

Projekt bauen

- Vorteile
 - Projekt kann aus IDE heraus gestartet/getestet werden
 - Schnell eingerichtet/verfügbar

Projekt bauen

- Nachteile
 - IDE muss installiert/eingerichtet sein
 - Benutzer-/Systemabhängig
 - Komplexe Builds problematisch
 - Non-headless
 - Potenziell fehleranfällig
 - Nicht automatisierbar

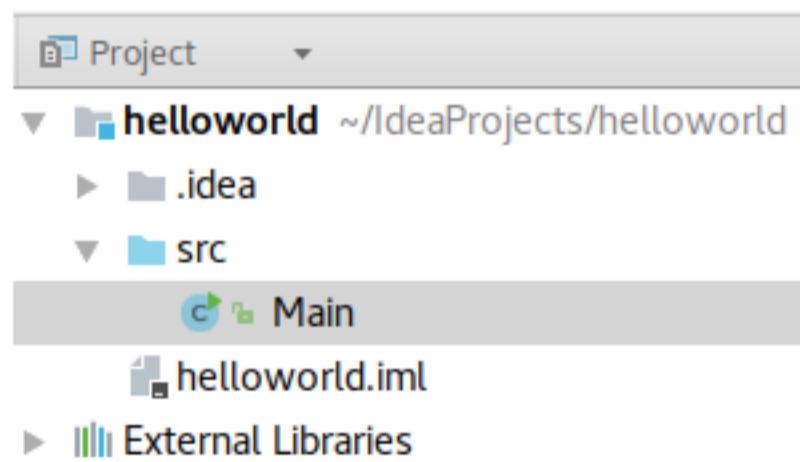
Projekt bauen

- Nachteile
 - Kein Dependency Management
 - Nicht pflegbar
 - Nur für Entwickler/Advanced User

Build und Deployment

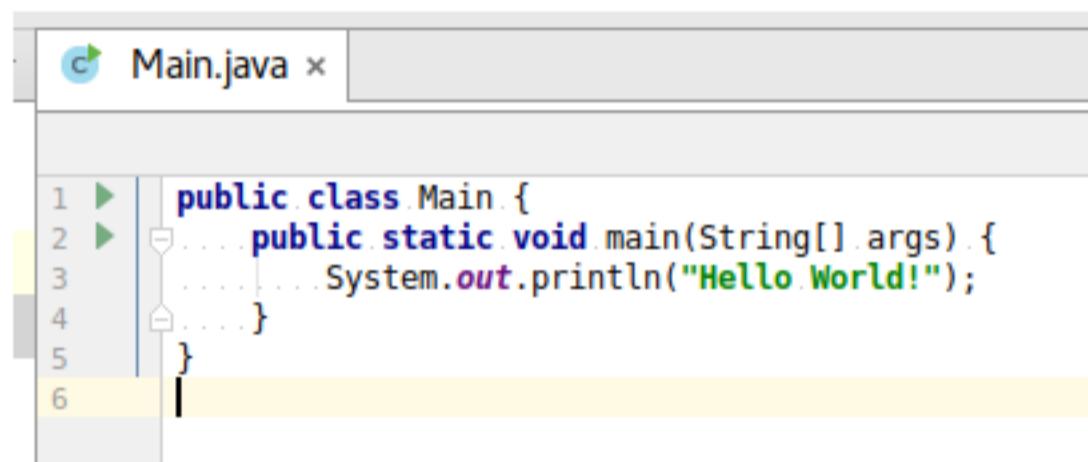
Projekt bauen

Projekt-Struktur



Projekt bauen

Code



A screenshot of a Java code editor showing a file named "Main.java". The code contains a single class definition:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4     }  
5 }  
6
```

Build und Deployment

Projekt bauen

Anwendung kompilieren und starten



Ergebnis



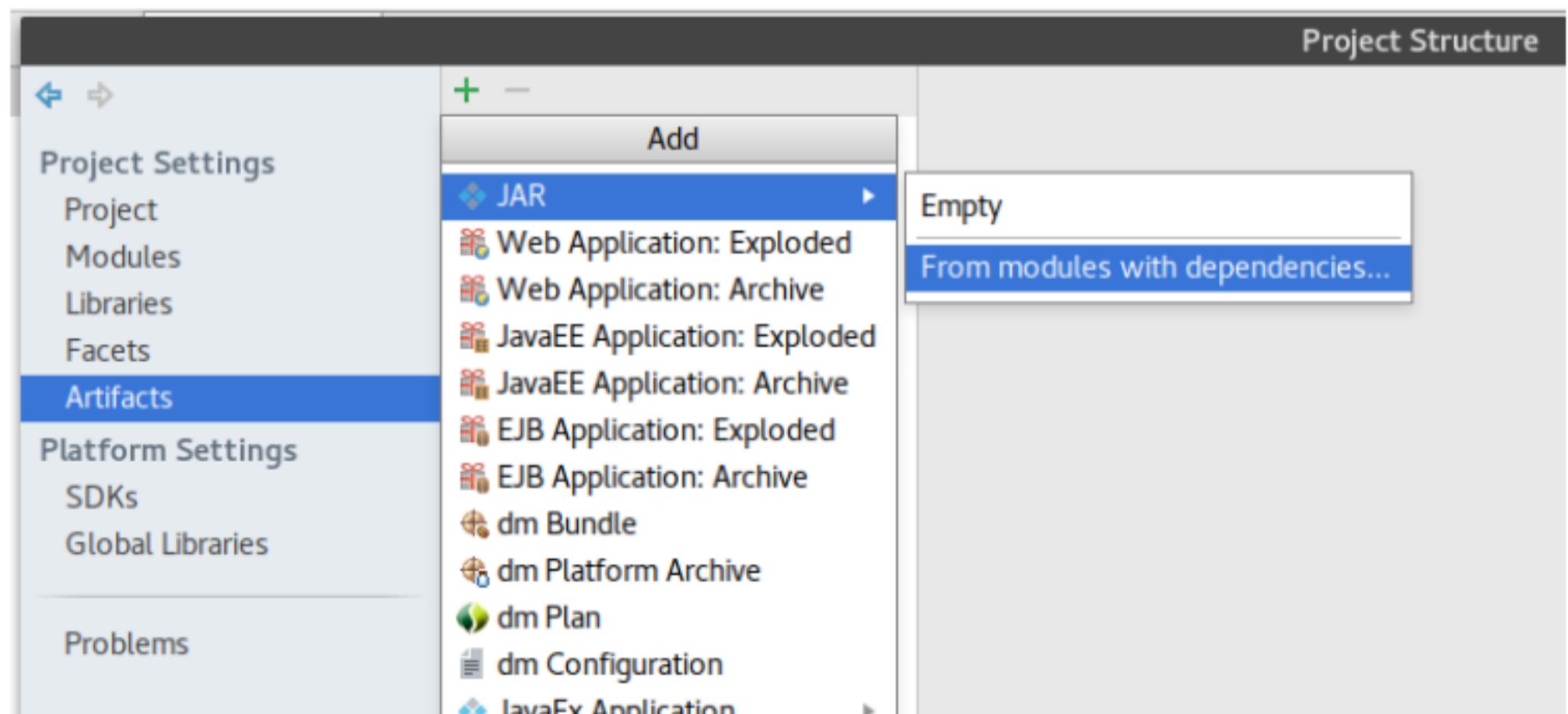
```
Main
/usr/lib/jvm/default/bin/java ...
Hello World!
```

Build und Deployment

Projekt bauen

JAR-Artefakt konfigurieren

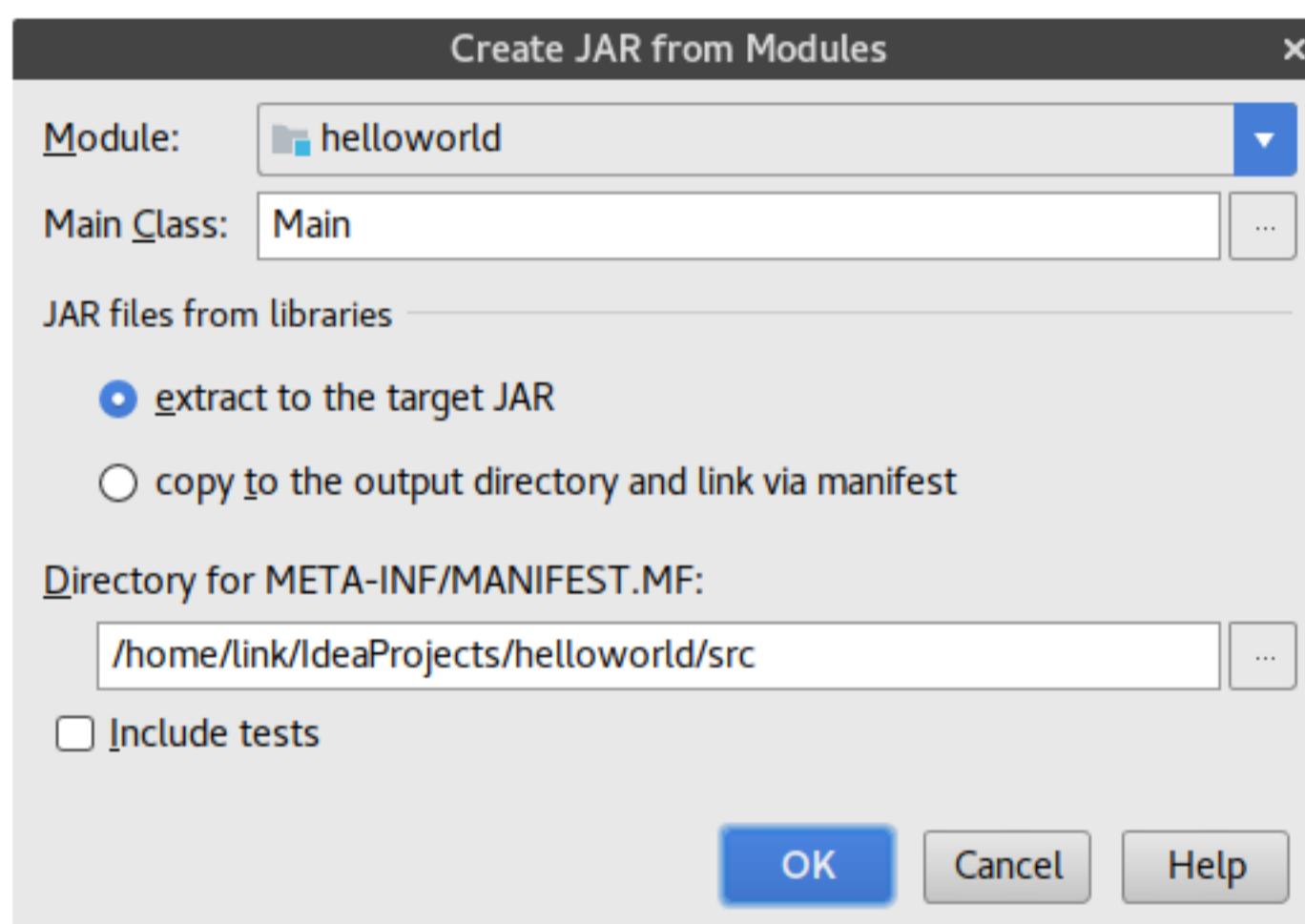
File -> Project Structure -> Artefacts -> Add ->
JAR



Build und Deployment

Projekt bauen

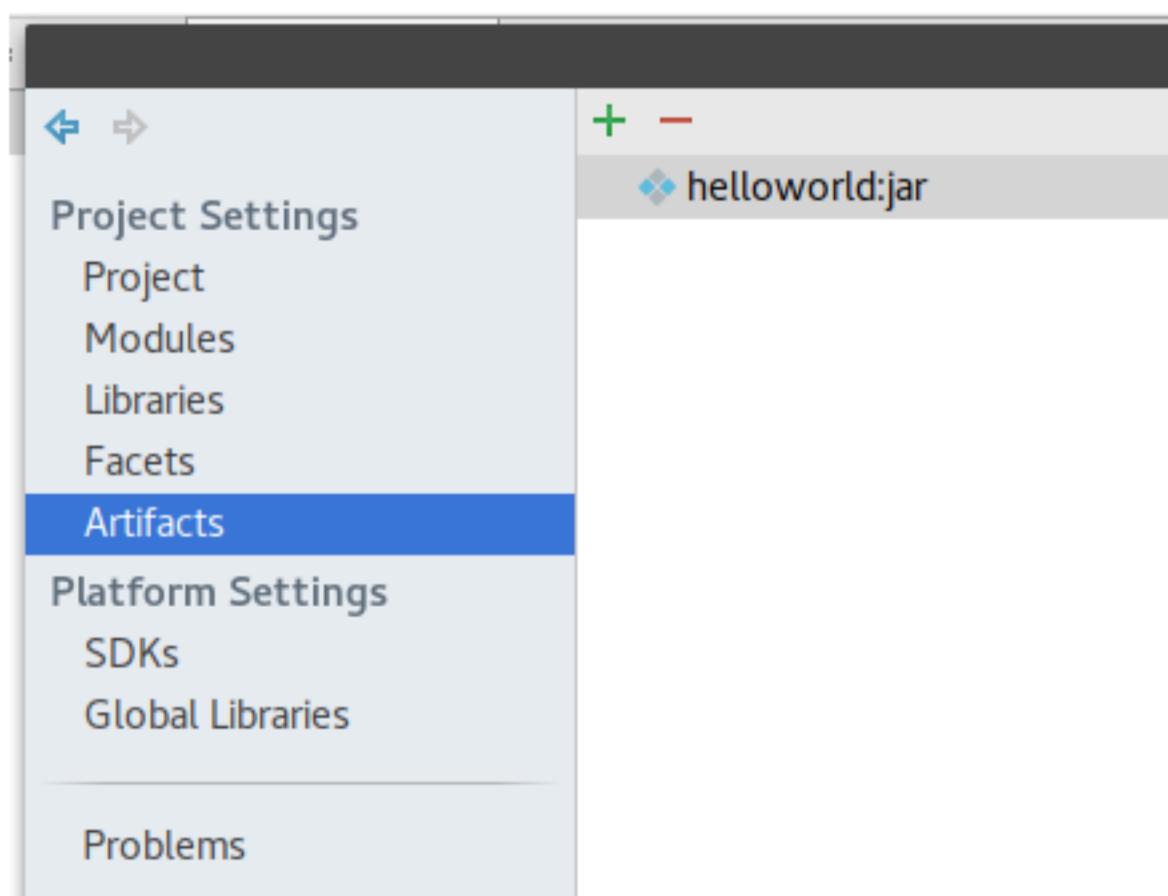
JAR-Artefakt konfigurieren



Build und Deployment

Projekt bauen

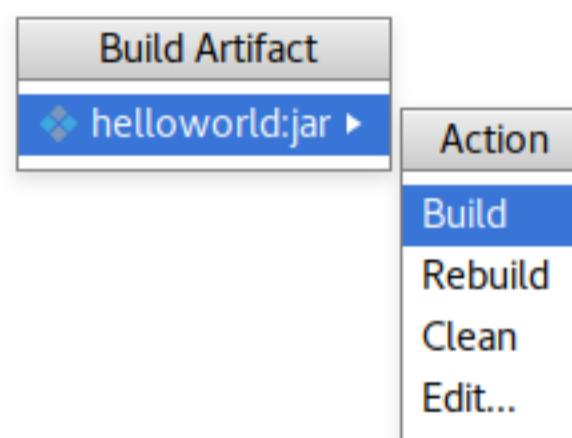
JAR-Artefakt konfigurieren



Build und Deployment

Projekt bauen

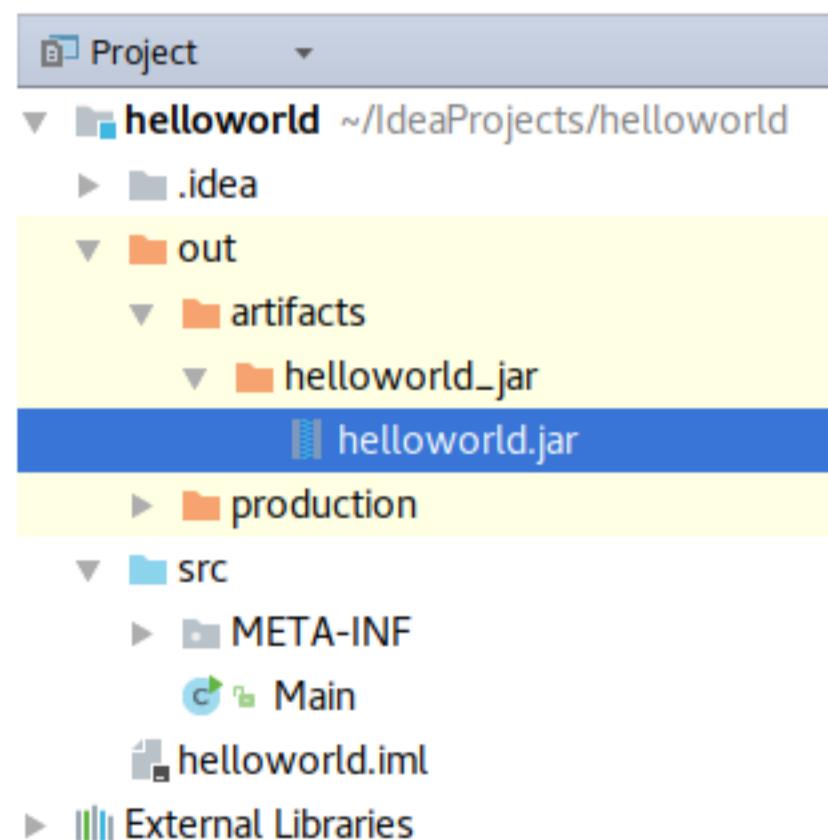
JAR-Artefakt bauen



Build und Deployment

Projekt bauen

JAR-Artefakt bauen



Build und Deployment

Projekt bauen

JAR-Artefakt ausführen

```
Terminal
+ link@arcangel helloworld_jar $ java -jar helloworld.jar
Hello World!
```

Build und Deployment

Projekt bauen

Variante 2: Skripte schreiben/verwenden

Projekt bauen

- Skript-Sprachen
 - Shell
 - Batch
 - PowerShell
 - Python
 - ...

Projekt bauen

- Vorteile
 - Simpel aufzurufen
 - Automatisierbar
 - Headless
 - Komplexe Builds möglich
 - Sehr flexibel

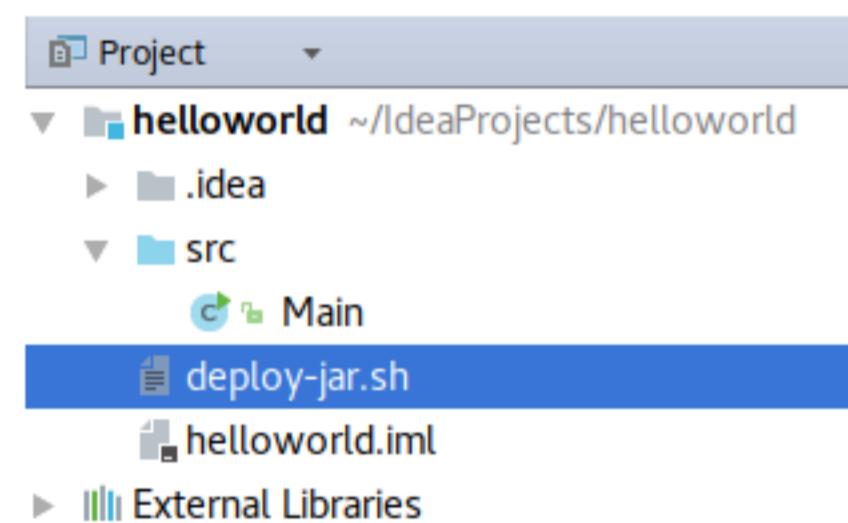
Projekt bauen

- Nachteile
 - Recht aufwendig (Erstellung/Pflege)
 - Einarbeitung nötig
 - CLI für Aufruf benötigt
 - Potenziell fehleranfällig
 - Kein Dependency Management
 - Systemabhängig

Build und Deployment

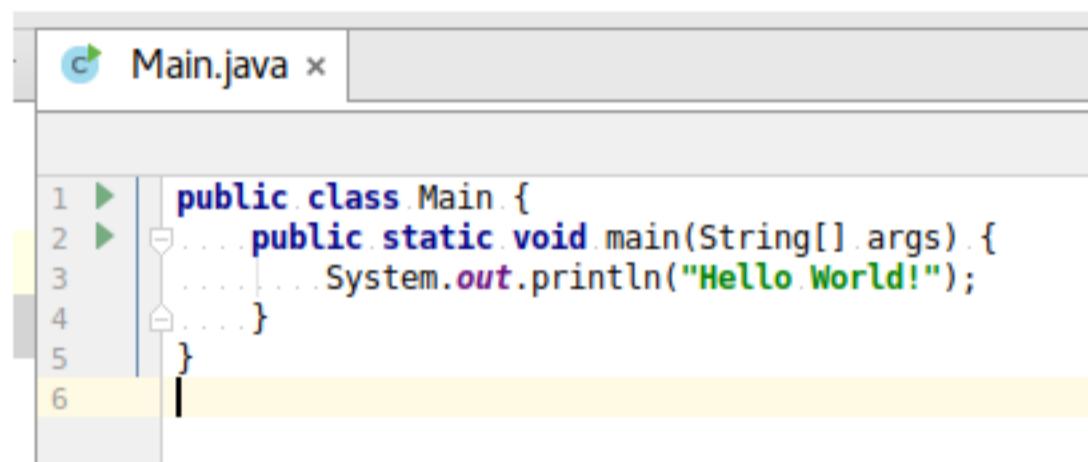
Projekt bauen

Projekt-Struktur



Projekt bauen

Code



A screenshot of a Java code editor showing a file named "Main.java". The code contains a single class definition:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4     }  
5 }  
6
```

Projekt bauen

Shell-Skript

```
deploy-jar.sh x
1  #!/bin/bash
2
3  mkdir -p out/classes
4
5  find ... -type f -name '*.java' | xargs javac -d out/classes
6
7  mkdir -p out/artefacts/helloworld_jar
8
9  jar -cfe out/artefacts/helloworld_jar/helloworld.jar Main -C out/classes ..
10
```

Build und Deployment

Projekt bauen

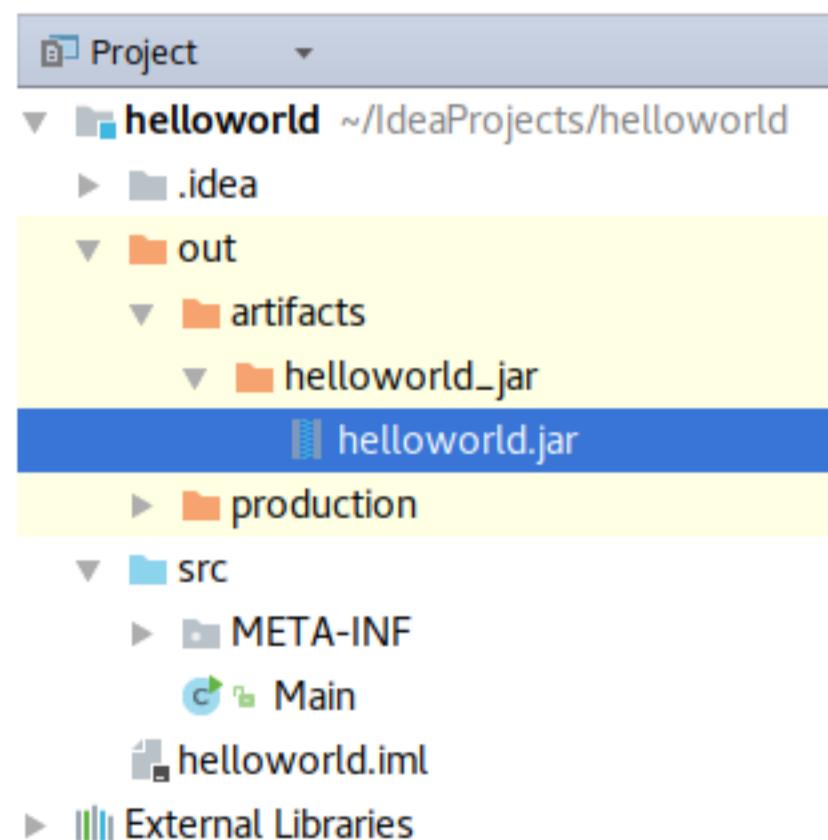
Shell-Skript ausführen

```
Terminal
+ link@arcangel helloworld $ ./deploy-jar.sh
x link@arcangel helloworld $
```

Build und Deployment

Projekt bauen

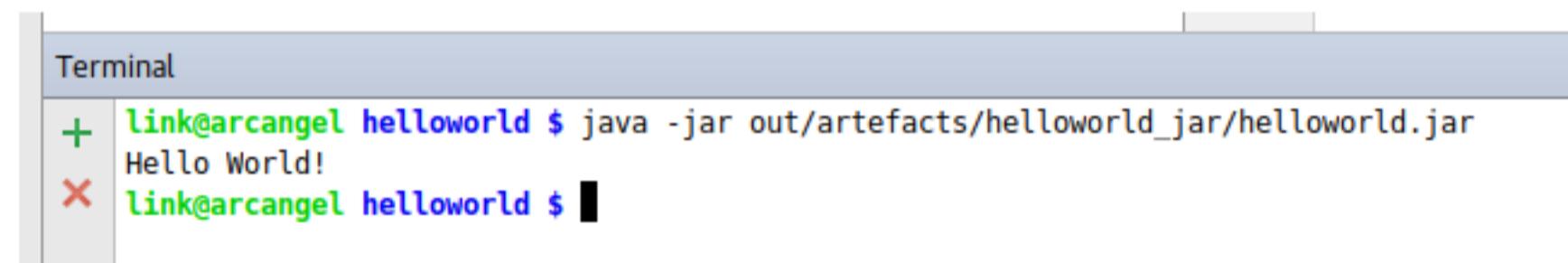
JAR-Artefakt



Build und Deployment

Projekt bauen

Shell-Skript ausführen



A screenshot of a terminal window titled "Terminal". The window shows a single command being run and its output. The command is "java -jar out/artefacts/helloworld_jar/helloworld.jar". The output is "Hello World!". The terminal has a standard OS X-style interface with a title bar and scroll bars.

```
+ link@garcangel helloworld $ java -jar out/artefacts/helloworld_jar/helloworld.jar
Hello World!
link@garcangel helloworld $
```

Build und Deployment

Projekt bauen

Variante 3: Build-Tools verwenden

Projekt bauen

- Build-Tools
 - Make
 - Ant
 - Maven
 - Gradle
 - MSBuild
 - ...

Projekt bauen

- Vorteile
 - Automatisierbar
 - Headless
 - "Nur" Konfiguration nötig
 - Systemunabhängig
 - Dependency Management
 - Komplexe Builds möglich

Projekt bauen

- Vorteile
 - Gut pflegbar/anpassbar
 - Viele zusätzliche Features/Plugins
 - IDE-Support

Build und Deployment

Projekt bauen

- Nachteile
 - Systemabhängig
 - Einarbeitung nötig

Projekt bauen – npm

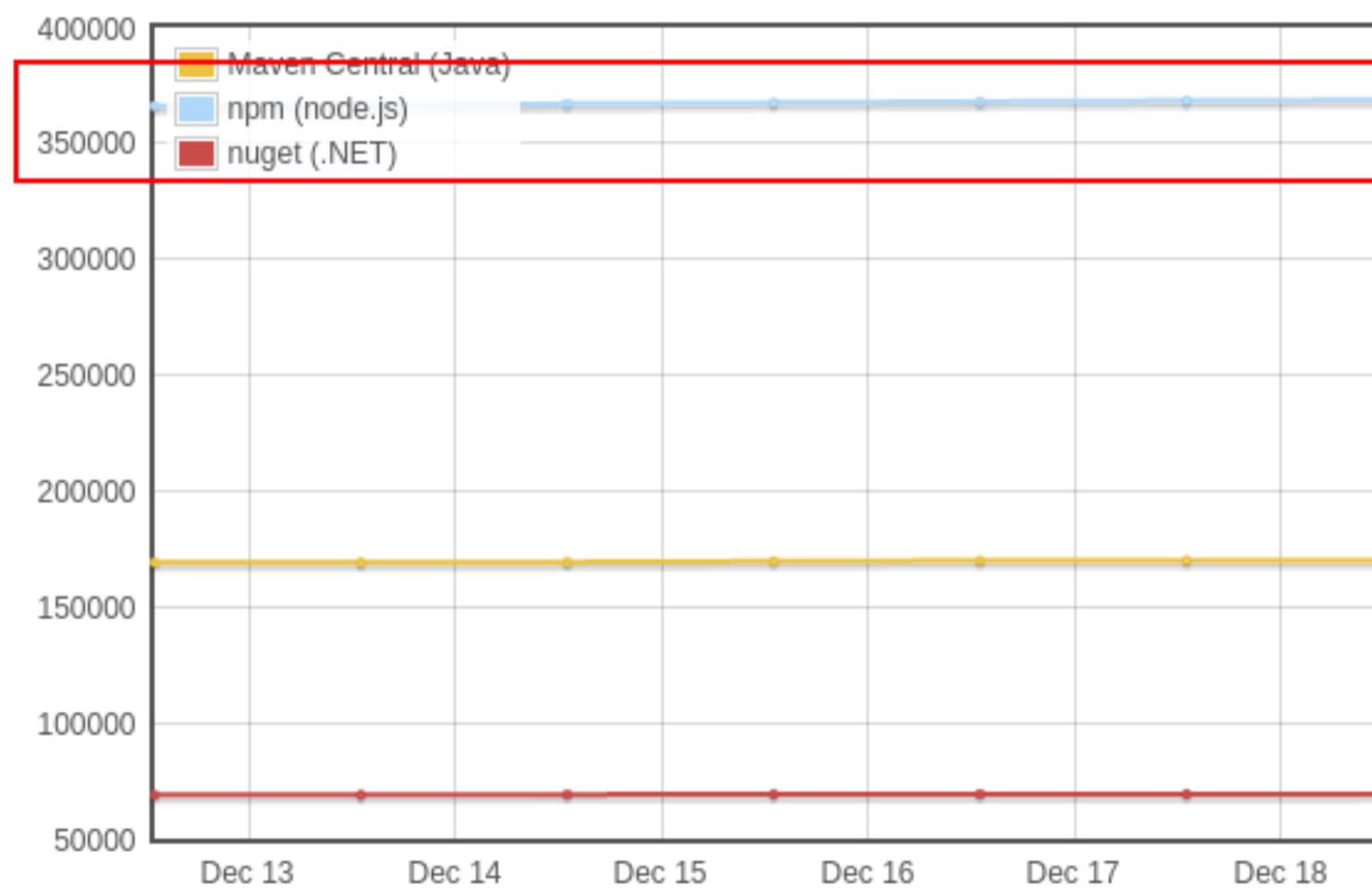
- Node Package Manager
- Script-Runner und Dependency Manager
- Gewaltiges Ökosystem (Packages/Module)
- Allgemein für JS-Projekte geeignet

Build und Deployment

Projekt bauen – npm

Anzahl npm-Packages

Module Counts



Build und Deployment

Projekt bauen – npm

npm-Projekt initialisieren

```
$ npm init
```

Projekt bauen – npm

Frage beantworten

```
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.
```

```
See `npm help json` for definitive documentation on these fields  
and exactly what they do.
```

```
Use `npm install <pkg> --save` afterwards to install a package and  
save it as a dependency in the package.json file.
```

```
Press ^C at any time to quit.
```

```
name: (mapeditor)  
version: (1.0.0)  
description: A mapeditor.  
entry point: (index.js)  
test command:  
git repository:  
keywords: mapeditor map editor  
author: Alexej, Karsten, Oxana, Ruben  
license: (ISC) UNLICENSED
```

Build und Deployment

Projekt bauen – npm

Ergebnis bestätigen

```
About to write to /home/link/develop/mapeditor/package.json:

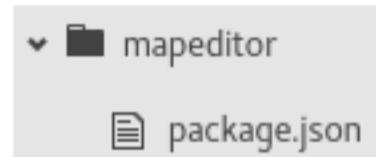
{
  "name": "mapeditor",
  "version": "1.0.0",
  "description": "A mapeditor.",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [
    "mapeditor",
    "map",
    "editor"
  ],
  "author": "Alexej, Karsten, Oxana, Ruben",
  "license": "UNLICENSED"
}

Is this ok? (yes) █
```

Build und Deployment

Projekt bauen – npm

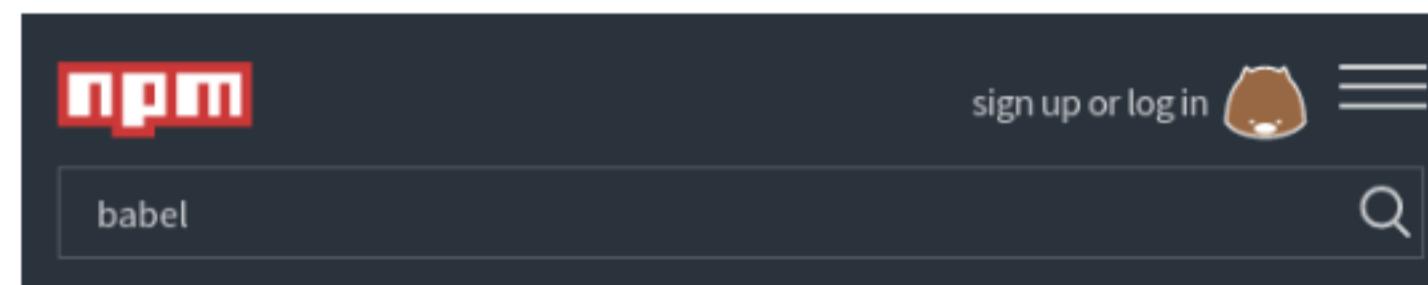
Erzeugte package.json



Build und Deployment

Projekt bauen – npm

npm-Pakete suchen



3102 results for 'babel'

Babel lucaswoj

Babel puts a soft cushion between you and all the cool new file formats being developed for node.js such as CoffeeScript, SASS, and Jade.

★ 20 v0.0.1

babel-loader danez

babel module loader for webpack

★ 86 v6.2.10

webpack, loader, babel, es6, transpiler, module

babel-eslint hzoo

Build und Deployment

Projekt bauen – npm

Neue Abhängigkeiten installieren

```
$ npm install -SE \
> leaflet jquery bootstrap babel-cli webpack
```

Benötigte Abhängigkeiten installieren

```
$ npm install
```

Build und Deployment

Projekt bauen – npm

Unsere package.json

```
{  
  "scripts": {  
    "build": "webpack",  
    "start": "webpack-dev-server --port 3000 --inline --watch --hot --open",  
    "doc": "jsdoc -r ./src/main/js"  
  },  
}
```

Projekt bauen – npm

Unsere package.json

```
.. "dependencies": {  
...   "babel": "6.5.2",  
...   "babel-cli": "6.18.0",  
...   "babel-loader": "6.2.10",  
...   "babel-preset-es2015": "6.18.0",  
...   "babel-preset-stage-0": "6.16.0",  
...   "file-loader": "0.9.0",  
...   "webpack": "1.14.0",  
...   "webpack-dev-server": "1.16.2",  
...   "bootstrap": "3.3.7",  
...   "copy-webpack-plugin": "4.0.1",  
...   "jquery": "3.1.1",  
...   "jsdoc": "3.4.3",  
...   "leaflet": "1.0.2",  
...   "q": "1.4.1"  
... }  
}
```

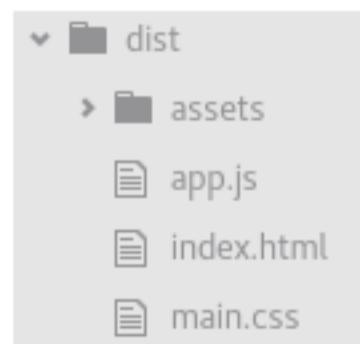
Build und Deployment

Projekt bauen – npm

Projekt bauen

```
$ npm run build
```

Ergebnis



Projekt bauen – Gradle

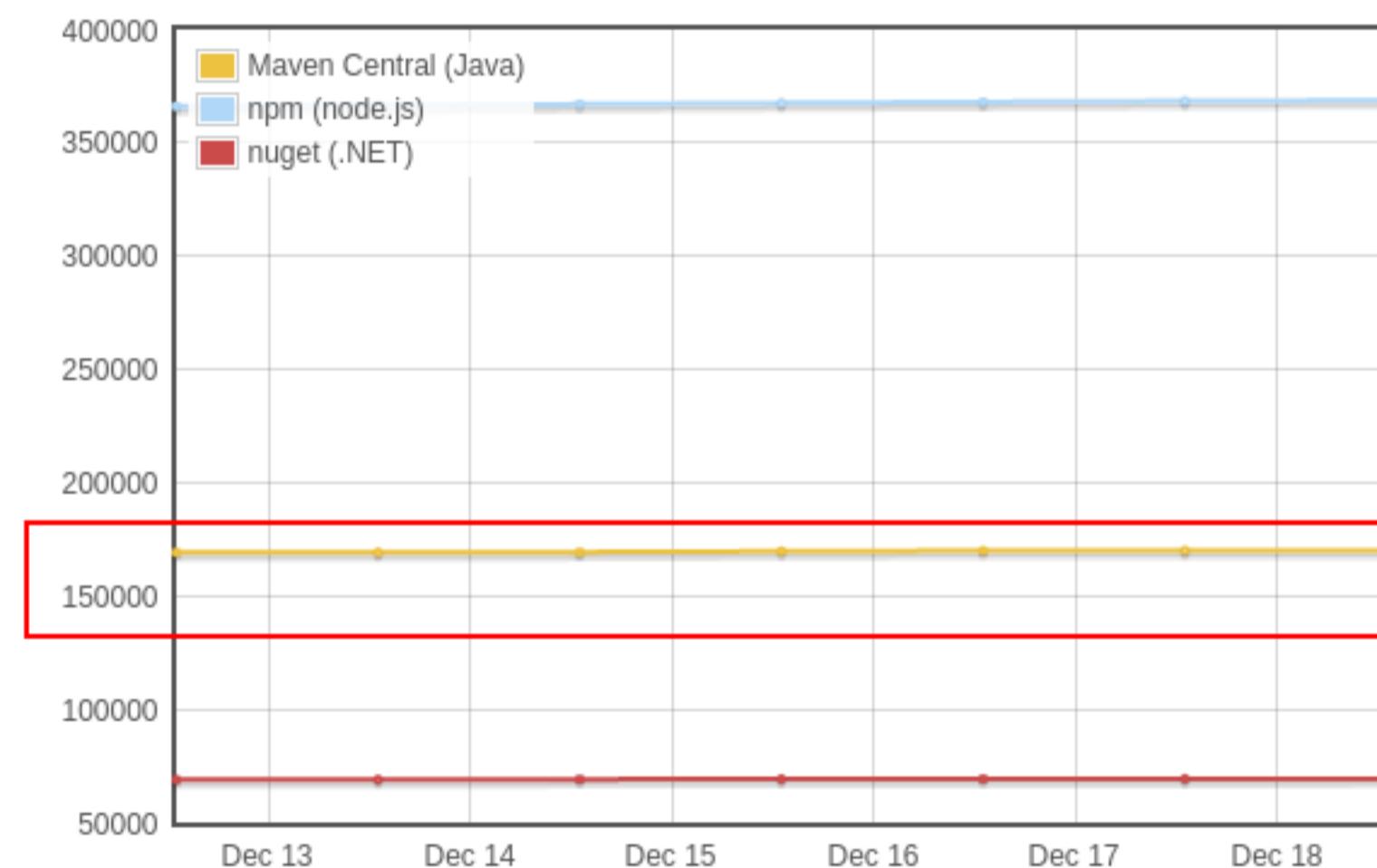
- Build-Tool und Dependency Manager
- Unterstützt polyglote Projekte
- Zugriff auf viele Repositories
- Sehr viele Plugins
- Gradle-Wrapper Skripte
- Bidirektonaler IDE-Support

Build und Deployment

Projekt bauen – Gradle

Anzahl Gradle-Packages

Module Counts



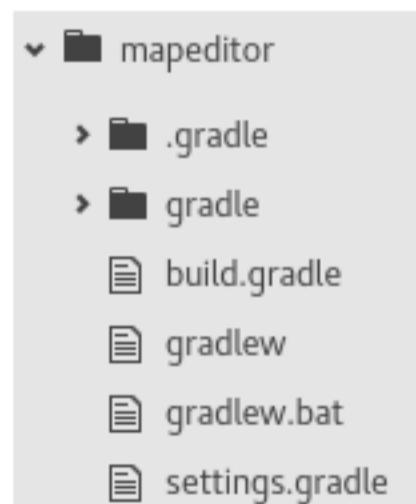
Build und Deployment

Projekt bauen – Gradle

Gradle-Projekt initialisieren

```
$ gradle init
```

Initialisiertes Gradle-Projekt



Projekt bauen – Gradle

settings.gradle

```
settings.gradle
1 rootProject.name = 'mapeditor'
2
```

Projekt bauen – Gradle

Generiertes build.gradle

```
build.gradle
1 apply plugin: 'java'
2
3 repositories {
4     jcenter()
5 }
6
7 dependencies {
8     compile 'org.slf4j:slf4j-api:1.7.21'
9
10    testCompile 'junit:junit:4.12'
11 }
```

Projekt bauen – Gradle

Unser build.gradle

```
build.gradle
1 apply plugin: "java"
2
3 apply plugin: "eclipse"
4 apply plugin: "idea"
5
6 apply plugin: "war"
7
8 version = "0.1-SNAPSHOT"
9
10 sourceCompatibility = 1.8
11
12 war {
13     archiveName = "tetraeder-obb-mapeditor.war"
14 }
15
```

Projekt bauen – Gradle

Unser build.gradle

```
15  ...
16  repositories { ...
17  |   mavenCentral() ...
18  }
19 ...
20 dependencies { ...
21 |   compile "javax:javaee-api:7.0" ...
22 |   compile "org.postgresql:postgresql:9.4.1211" ...
23 |
24 |   compile "commons-io:commons-io:2.5" ...
25 |   compile 'org.apache.commons:commons-lang3:3.5' ...
26 |
27 |   compile "com.google.code.gson:gson:2.8.0" ...
28 |
29 |   testCompile "junit:junit:4.12" ...
30 }
```

Build und Deployment

Projekt bauen – Gradle

Abhängigkeiten finden

java ee api

Found 26366 results



1. Java(TM) EE 7 Specification APIs
[javax » javaee-api](#)

Java(TM) EE 7 Specification APIs



2. Java EE: EJB API V3.0
[org.ow2.spec.ee » ow2-ejb-3.0-spec](#)

EJB 3 API of Java EE 5.0

Build und Deployment

Projekt bauen – Gradle

Abhängigkeiten finden

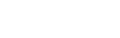
[Home](#) » [javax](#) » [javaee-api](#)



Java(TM) EE 7 Specification APIs

Java(TM) EE 7 Specification APIs

License	CDDL 2 GPL
Categories	Java Specifications
Tags	standard javaee javax api specs
Used By	738 artifacts

	Version	Repository	Usages	Date
7.0.x	7.0	Central	 407	(May, 2013)
	6.0	Central	 355	(Jan, 2010)
6.0.x	6.0-RC2	Central	 0	(May, 2010)
	6.0-RC1	Central	 0	(May, 2010)

Build und Deployment

Projekt bauen – Gradle

Abhängigkeiten finden

[Home](#) » [javax](#) » [javaee-api](#) » 7.0



Java(TM) EE 7 Specification APIs » 7.0

Java(TM) EE 7 Specification APIs

License	CDDL 2 GPL
Categories	Java Specifications
Date	(May 20, 2013)
Files	Download (JAR) (4.4 MB)
Repositories	Central Java.net Releases
Used By	738 artifacts

[Maven](#) [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
compile group: 'javax', name: 'javaee-api', version: '7.0'
```

Build und Deployment

Projekt bauen – Gradle

IDE-Konfiguration generieren

```
$ ./gradlew idea
```

IDE-Konfiguration generieren

- 📄 mapeditor.iml
- 📄 mapeditor.ipr
- 📄 mapeditor.iws

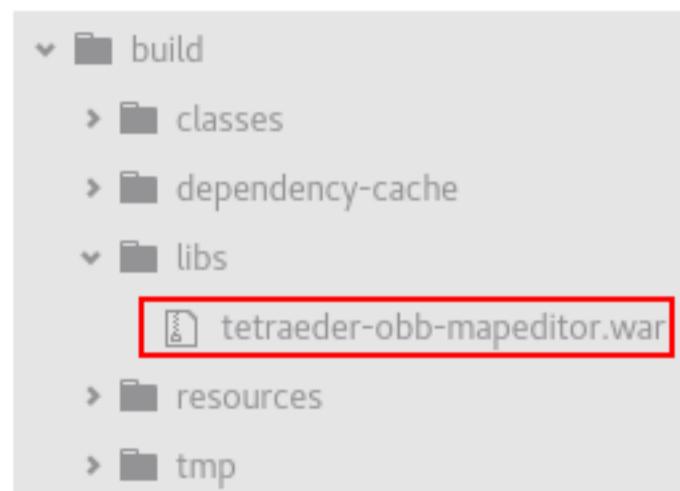
Build und Deployment

Projekt bauen – Gradle

WAR-Archiv bauen

```
$ ./gradlew war
```

Fertiges WAR-Archiv



Systemabhängigkeit eliminieren

Warum?

- Jedes System ist anders
- Ziel-System relevant

Systemabhängigkeit eliminieren

Ziel

- Bestmöglicher Ausschluss von Umweltfaktoren
- Bestmöglicher Nachbau des Ziel-Systems

=> "Works On My Machine" verhindern

=> Sandkasten

Systemabhängigkeit eliminieren

Möglichkeiten

- Virtuelle Maschinen
- Container

Virtuelle Maschinen

Vorteile

- Sandbox-Wirkung
- Nach Bedarf verwendbar

Virtuelle Maschinen

Nachteile

- Einrichtung
- Performance
- Speicherverbrauch
- Prozesse von außen nicht überwachbar

Virtuelle Maschinen

Kleines Helferlein: vagrant

- Plattformunabhängig
- Support für verschiedene Hypervisor
- Fertig installierte Images in Repositories
- VM-Konfiguration/-Einrichtung via Skript (Vagrantfile)
- Einrichtung automatisierbar/reproduzierbar

Container (mit Docker)

- Leichtgewichtige virtuelle Umgebungen (keine VMs!)
- Image-Konfiguration via Textdatei (Dockerfile)
- Container basieren auf Image
- Prozesse laufen nativ

Container (mit Docker)

Vorteile

- Sandbox-Wirkung
- Nahezu kein Overhead
- Sehr performant
- Speicherverbrauch nur durch enthaltene Anwendung
- Container-Erzeugung dauert keine 5 Sekunden

Container (mit Docker)

Vorteile

- Anwendungen unterschiedlicher Versionen gleichzeitig ausführen
- Prozesse von außen überwachbar
- Zu größeren Systemen kombinierbar
- Built-in Cluster-Support
- ...

Container (mit Docker)

Nachteile

- Einarbeitung
- Oft als Pseudo-VM misbraucht

Container (mit Docker)

Container starten (Hello World)

```
$ hostname  
arcangel  
$ docker run --rm ubuntu \  
> sh -c 'echo "Ich bin der Container: $(hostname)"'  
Ich bin der Container: b318447dae26
```

Build und Deployment

Container (mit Docker)

Container starten (PostgreSQL)

```
$ docker run -d -p 5432:5432 \
> -v "$PWD"/postgres-data:/var/lib/postgresql/data \
> postgres:9.3.11
```

Container (mit Docker)

Eigenes Image bauen

Dockerfile

```
1 FROM postgres:9.3.11
2
3 RUN apt-get update && \
4     apt-get install -y \
5         dos2unix && \
6     rm -rf /var/lib/apt/lists/*
7
8 COPY sql/* /docker-entrypoint-initdb.d/
9
10 RUN for i in /docker-entrypoint-initdb.d/* ; do \
11     dos2unix $i ; \
12     done
```

Build und Deployment

Container (mit Docker)

Eigenes Image bauen

```
$ docker build -t mein-postgres-image .
```

Build und Deployment

Container (mit Docker)

Eigenes Image bauen

```
Sending build context to Docker daemon 9.728 kB
Step 1 : FROM postgres:9.3.11
--> 24a3326f0c9e
Step 2 : RUN apt-get update &&      apt-get install -y      dos2unix &&      rm
-rf /var/lib/apt/lists/*
--> Using cache
--> 0d5709c1c449
Step 3 : COPY sql/* /docker-entrypoint-initdb.d/
--> 45915c0bb921
Removing intermediate container 9fd4212e039d
Step 4 : RUN for i in /docker-entrypoint-initdb.d/* ; do      dos2unix $i ;
done
--> Running in 9324ee5a44ae
dos2unix: converting file /docker-entrypoint-initdb.d/00-create-schema.sql to Unix format ...
dos2unix: converting file /docker-entrypoint-initdb.d/01-dummy-content.sql to Unix format ...
--> eda8c5fad615
Removing intermediate container 9324ee5a44ae
Successfully built eda8c5fad615
```

Container (mit Docker)

Container vom eigenen Image erstellen

```
$ docker run -d -p 5432:5432 \
> -v "$PWD"/postgres-data:/var/lib/postgresql/data \
> mein-postgres-image
```

Build und Deployment

Container (mit Docker)

Leben erleichtern mit Docker Compose

```
docker-compose.yml
1 version: "2"
2
3 services:
4   db:
5     build: db
6     ports:
7       - "15432:5432"
8   appserver:
9     build: appserver
10    depends_on:
11      - db
12    links:
13      - db
14    ports:
15      - "18080:8080"
16      - "14848:4848"
17    volumes:
18      - ./backend/build/libs:/opt/payara41/glassfish/domains/domain1/autodeploy
```

Build und Deployment

Container (mit Docker)

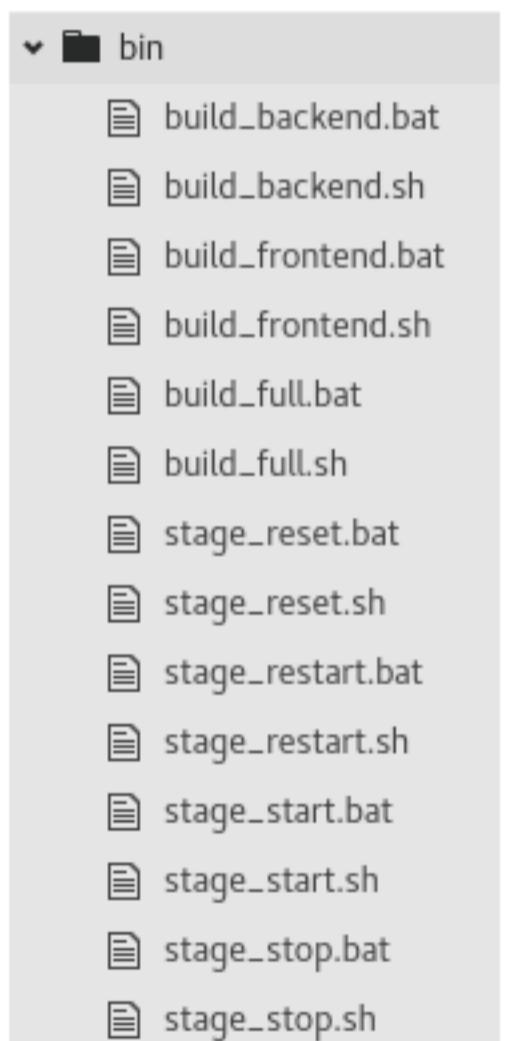
Leben erleichtern mit Docker Compose

```
$ docker-compose up -d appserver
```

Build und Deployment

Container (mit Docker)

Und noch einfacher ...



Container (mit Docker)

Shell-Skript

```
build_frontend.sh
1 #!/bin/bash
2
3 export WORKER_UID=$(id -u)
4
5 cd $(dirname $(realpath $0))/..
6
7 docker-compose up --build build-frontend
8
```

Container (mit Docker)

Batch-Skript

build_frontend.bat

```
1 @echo off
2 ^
3 cd "%~dp0"..
4 ^
5 docker-compose up --build build-frontend
6
```

Build und Deployment

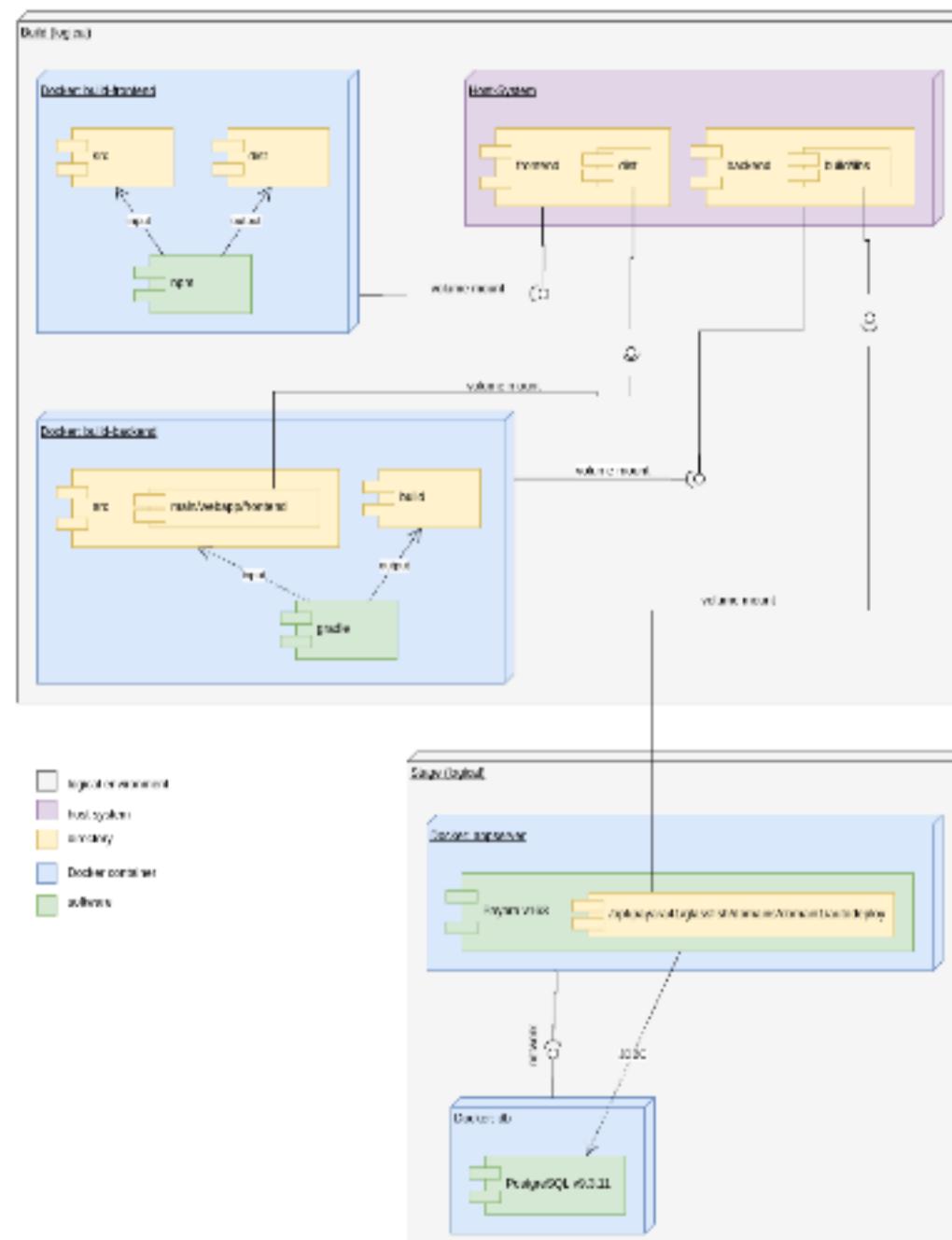
Container (mit Docker)

Shell-Skript ausführen

```
$ ./bin/build_frontend.sh █
```

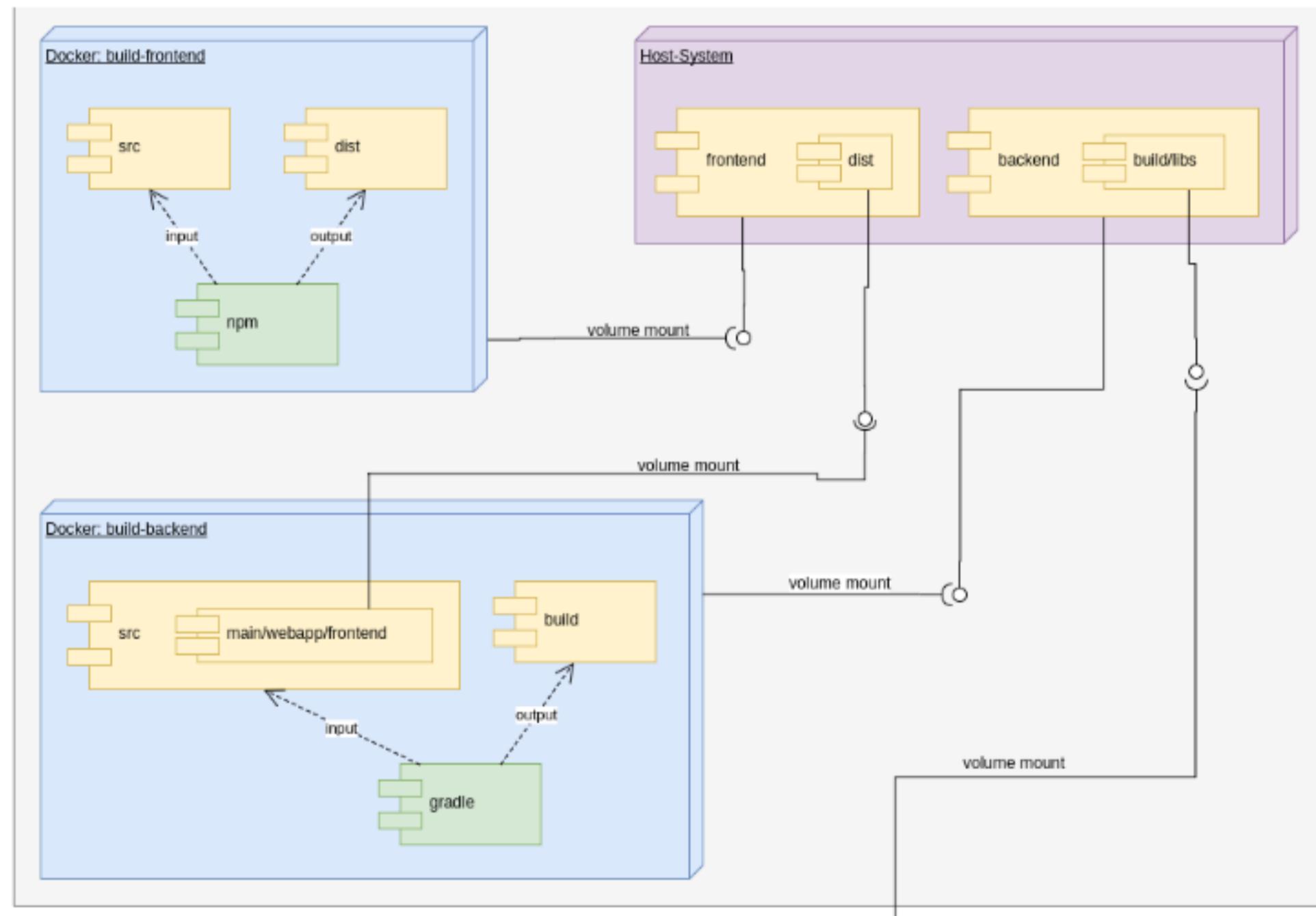
Build und Deployment

Unsere Build- und Stage-Umgebung



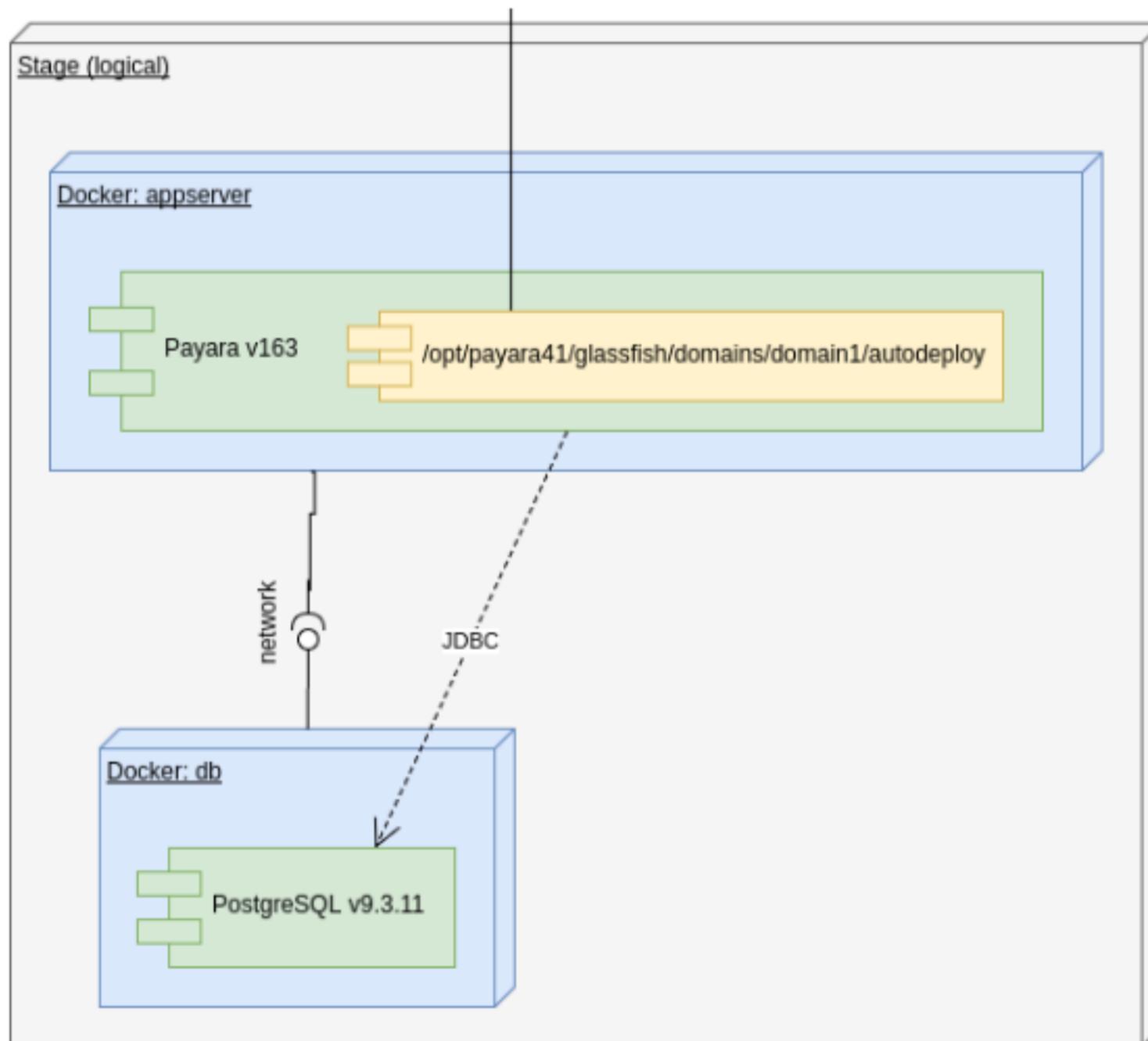
Build und Deployment

Unsere Build- und Stage-Umgebung



Build und Deployment

Unsere Build- und Stage-Umgebung



Zusammenfassung

- IDEs ungeeignet für reproduzierbare Builds
- Build-Tools lohnen sich
 - Dependency Management
 - Automatisierung
 - Reproduzierbar
 - Headless

Build und Deployment

Zusammenfassung

- Build-Umgebung dem Ziel-System nachempfinden
- Virtuelle Maschinen umständlich in Handhabung
- Container sind leichtgewichtige Alternative zu VMs

[Übersicht](#)

Ergebnis: Software Demo

Software Demo

Übersicht