

Erstellung einer Schnittstelle zur Steuerung von Lagerliften



Projektpräsentation von
Yannik Engel

Agenda



Vorstellung Unternehmen und Projekt



Analysephase



Entwurfsphase



Implementierungsphase



Fazit und Ausblick

Das Unternehmen

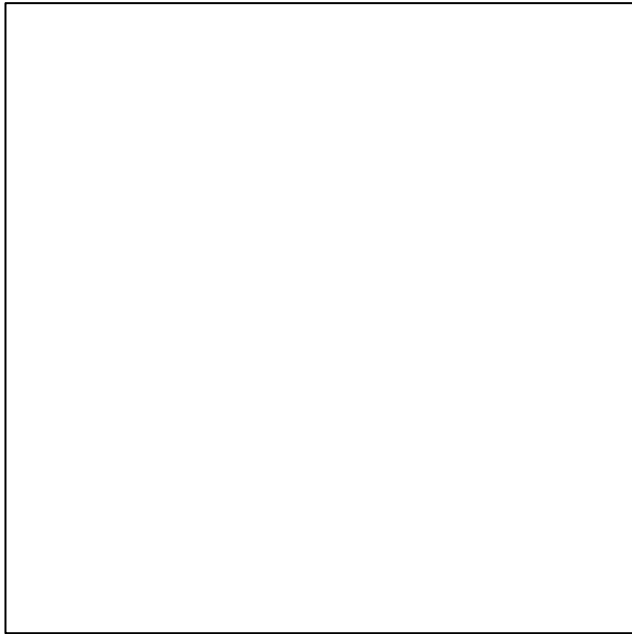


Das Projekt



- Automatisierung Lagerlifte
-
- Bedienung vereinfachen
 - Fehlerquellen verringern
 - Machine-to-Machine Kommunikation ermöglichen

Projektphasen



9 h

Gesamt: 70 Stunden

Agenda



Vorstellung Unternehmen und Projekt



Analysephase



Entwurfsphase



Implementierungsphase



Fazit und Ausblick

Projektkosten

Vorgang	Mitarbeiter	Zeitaufwand	Kosten
Entwicklungskosten	1 x Auszubildender	70 h	1591,10 €
Tests am Lift	2 x Mitarbeiter	6 h	300,00 €
Fachgespräche	1 x Mitarbeiter	3 h	120,00 €
Code Review	1 x Mitarbeiter	1 h	40,00 €
Gesamtkosten			2051,10 €

Auszubildender / Stunde: 7,73 €

Mitarbeiter / Stunde: 25,00 €

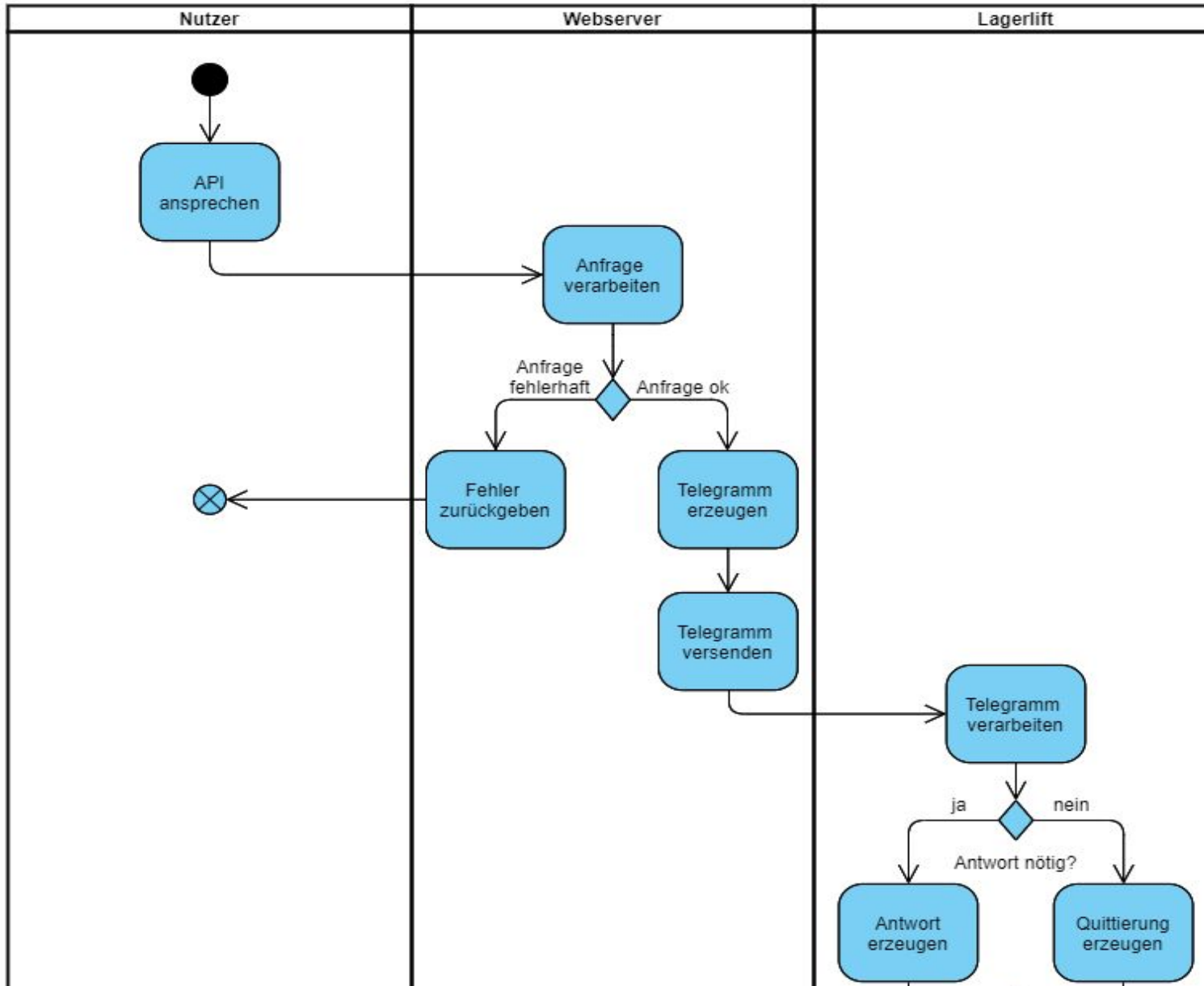
Ressourcen / Stunde: 15,00 €

Amortisation

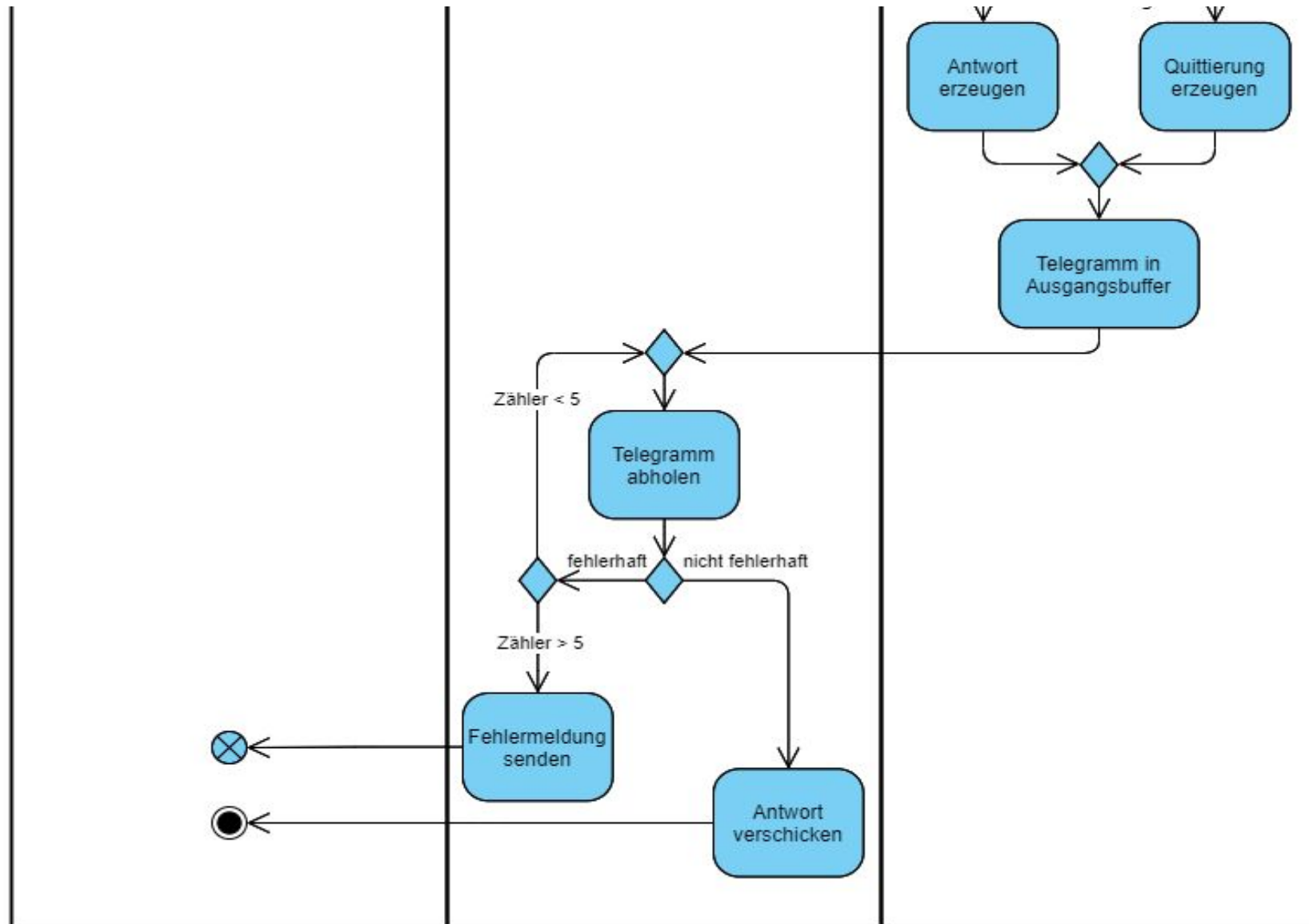
Vorgang	Anzahl pro Tag	Differenz pro Vorgang	Zeiteinsparung
Kommissionierung	6 x	7,5 min	45 min
Allg. Bedienung	10 x	0,5 min	5 min
Fehlervermeidung	1 x	10 min	10 min
Gesamteinsparung pro Tag			60 min



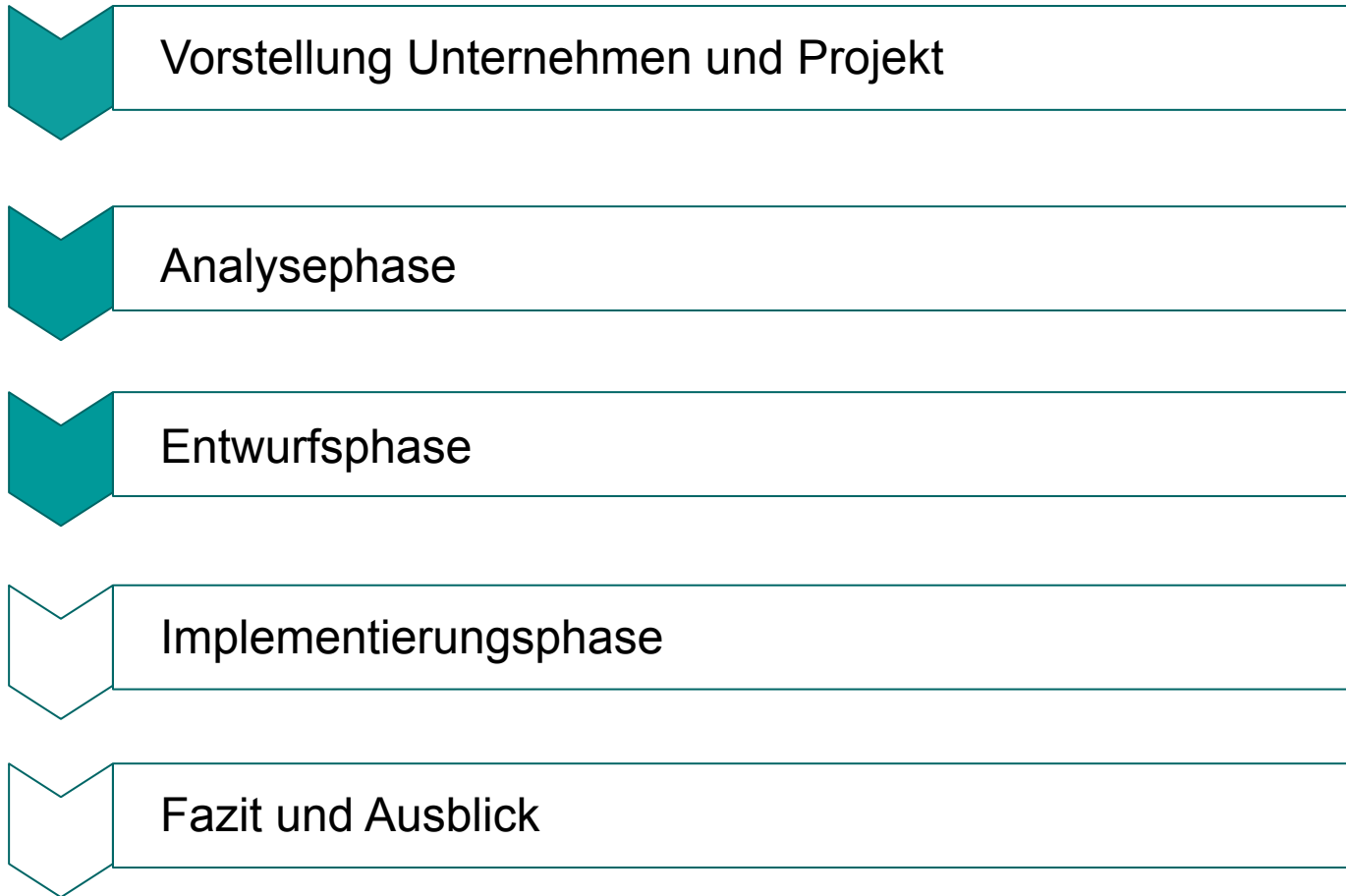
Aktivitätsdiagramm



Aktivitätsdiagramm



Agenda



Komponenten

Machine Interface

- express.js Webserver
- Middleware
- Implementierung mraa Library



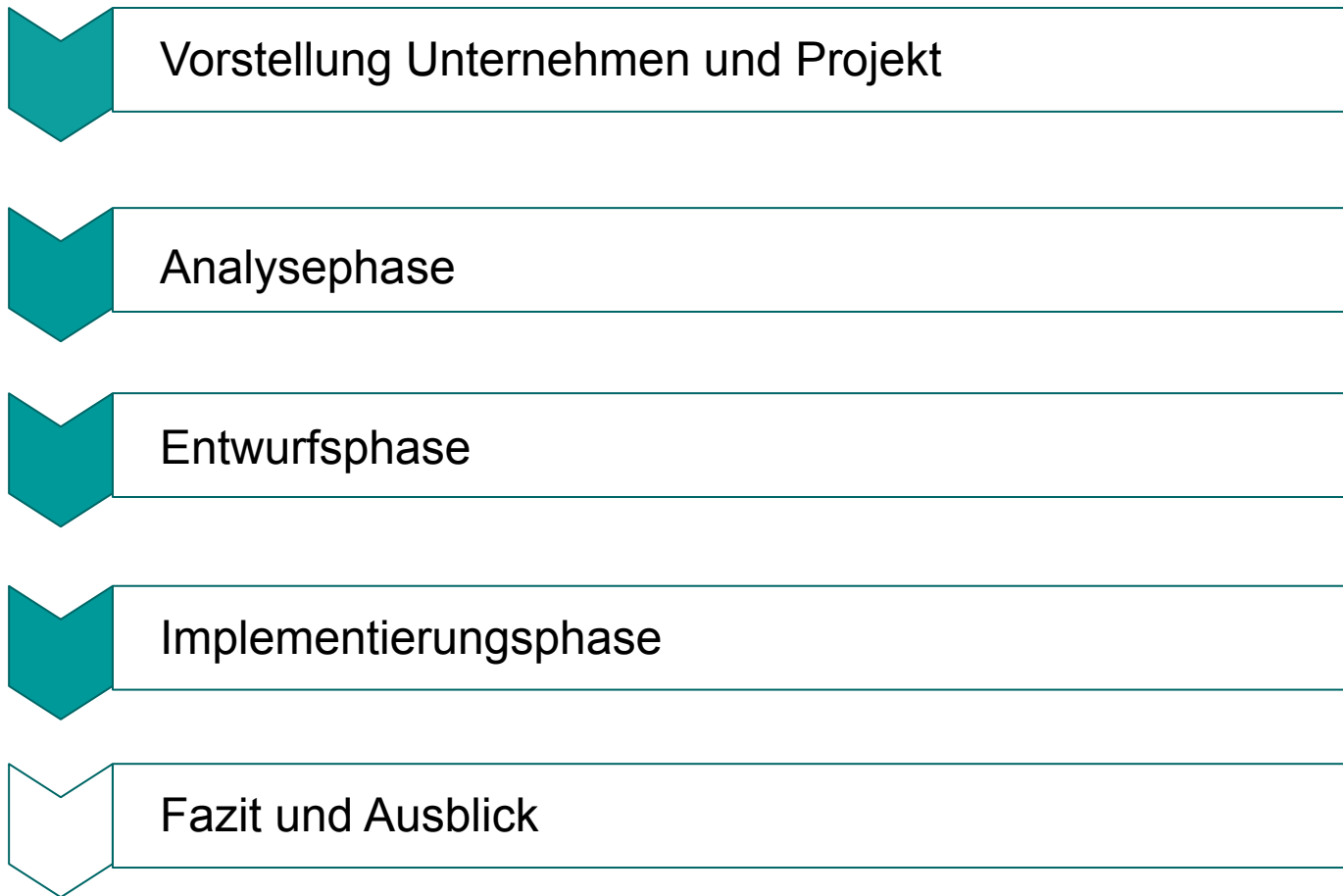
Interagieren

GUI

- Integration in Phytect Webtools
- Administrationsseite



Agenda



GUI

Artikelsuche

CV057-000000000



Fach ID	Fachtyp	Anzahl Rollen	Rollen Code	Artikelnummer	Hersteller	Menge	Einheit
3.4.2	Rollenständer klein	142	R200911-1110-011	CV057	0000000000	0.0	Stck.
3.4.2	Rollenständer klein	142	R200911-1110-013	CV057	0000000000	0.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-070	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-069	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-068	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-067	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-160	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-159	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-158	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-157	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-156	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-155	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-154	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-153	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-152	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-151	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-150	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-149	CV057	0000000000	50000.0	Stck.
50.1.1	Produktionshalle Ebene 2	2717	R200430-1111-148	CV057	0000000000	50000.0	Stck.

AUSLAGERN

GUI

Wähle Lift:



Fifo Speicher

Anzahl Befehle im Handspeicher

0

 LOESCHE SPEICHER

Füllstand

Aktuelle Informationen zum Füllstand

Füllgrad: %

Freie Lagerhöhe: mm


Maximale Lagerhöhe: mm

Sende Text

Zeile



Text

 ABSENDEN


Initialisation

Anzahl Displayzeilen



Wähle Eingabezeile



 INITIALISIERE TERMINAL

Steuersequenz

Wähle verschiedene Steuerbefehle

Aktion



 SENDE STEUERBEFEHL

Router

```
router.post('/start', function (req, res, next) {  
  
  let adresse = req.body.adresse;  
  let fachboden = req.body.fachboden;  
  let stellplatz = req.body.stellplatz;  
  let tiefe = req.body.tiefe;  
  let stringident = req.body.stringident;  
  let text1 = req.body.text1;  
  let text2 = req.body.text2;  
  
  if (log_level >= 3) {  
    console.log(adresse);  
    console.log(fachboden);  
    console.log(stellplatz);  
    console.log(tiefe);  
    console.log(stringident);  
    console.log(text1);  
    console.log(text2);  
  }  
  
  let result = mm260.start(adresse, fachboden, stellplatz, tiefe, stringident, text1, text2, sofort: false);  
  
  res.status(200).send(result);  
  
});
```

Einstieg Middleware

Middleware

```
function start(fachboden, stellplatz, tiefe, stringident, text1, text2, sofort) {

  if (!Number.isInteger(fachboden) || fachboden < 0 || fachboden > 99999) {
    throw new createError(400, 'Fachboden muss eine positive, ein- bis fünfstellige Zahl sein!');
  }

  let befehl = (sofort) ? '05' : '10';
  let string = buildStartString(fachboden, stellplatz, tiefe, stringident, text1, text2, befehl);

  if (string.length !== 66) {
    throw new createError(503, 'Telegramm hat falsche Länge');
  }

  conStop(); // Stoppt kontinuierliches Pollingfunktion
  let tries = 5; // Abbruchszähler

  while (tries > 0) {
    let telegram = uart.setDataGetData(string);
    let result = verarbeiteTelegram(telegram, filter_befehl: '30');
    conStart(); // Startet Pollingfunktion wieder

    if (Object.keys(result).length !== 0) {
      let validierung = validiereQuittierung(stringident, result);
      if (validierung.error === true) {
        throw new createError(503, validierung.message);
      }
      return JSON.stringify(validierung);
    }
    tries--;
  }
  throw new createError(503, 'Keine (korrekte) Antwort erhalten.');
```

Aufruf Hardware Service



mraa Implementierung

```
let mraa = require('mraa'); // Einbindung Hardware Library
uart = new mraa.Uart(port); // Erzeugung Kommunikationsobjekt

function getData(boolSleep = false) {
  let tmp = '';
  let char = '';

  if (boolSleep) {
    while (uart.dataAvailable()) { // Solange Daten vorhanden
      char = uart.readStr(1); // Zeichen einlesen
      tmp += char;
      if (log_level >= 3) {
        console.log(encodeURIComponent(tmp));
      }
    }
  } else {
    while (char !== stoppsequenz && uart.dataAvailable(data_read_timeout)) {
      char = uart.readStr(1);
      tmp += char;
      if (log_level >= 3) {
        console.log(encodeURIComponent(tmp));
      }
    }
  }
  console.log('Data read: ' + encodeURIComponent(tmp));
  return tmp; // Rueckgabe kompletter String
}
```

Middleware

```
function start(fachboden, stellplatz, tiefe, stringident, text1, text2, sofort) {  
  
    if (!Number.isInteger(fachboden) || fachboden < 0 || fachboden > 99999) {  
        throw new createError(400, 'Fachboden muss eine positive, ein- bis fünfstellige Zahl sein!');  
    }  
  
    let befehl = (sofort) ? '05' : '10';  
    let string = buildStartString(fachboden, stellplatz, tiefe, stringident, text1, text2, befehl);  
  
    if (string.length !== 66) {  
        throw new createError(503, 'Telegramm hat falsche Länge');  
    }  
  
    conStop(); // Stoppt kontinuierliches Pollingfunktion  
    let tries = 5; // Abbruchszähler  
  
    while (tries > 0) {  
        let telegram = uart.setDataGetData(string);  
        let result = verarbeiteTelegram(telegram, filter_befehl: '30');  
        conStart(); // Startet Pollingfunktion wieder  
  
        if (Object.keys(result).length !== 0) {  
            let validierung = validiereQuittierung(stringident, result);  
            if (validierung.error === true) {  
                throw new createError(503, validierung.message);  
            }  
            return JSON.stringify(validierung);  
        }  
        tries--;  
    }  
    throw new createError(503, 'Keine (korrekte) Antwort erhalten.');
```

Verarbeitung und
Quittierung

Rückgabe JSON-String

Router

```
router.post('/start', function (req, res, next) {  
  
  let adresse = req.body.adresse;  
  let fachboden = req.body.fachboden;  
  let stellplatz = req.body.stellplatz;  
  let tiefe = req.body.tiefe;  
  let stringident = req.body.stringident;  
  let text1 = req.body.text1;  
  let text2 = req.body.text2;  
  
  if (log_level >= 3) {  
    console.log(adresse);  
    console.log(fachboden);  
    console.log(stellplatz);  
    console.log(tiefe);  
    console.log(stringident);  
    console.log(text1);  
    console.log(text2);  
  }  
  
  let result = mm260.start(adresse, fachboden, stellplatz, tiefe, stringident, text1, text2, sofort: false);  
  
  res.status(200).send(result); ← Rückgabe an Aufrufer  
  
});
```

Testen

Tests im Büro

- Webserver Kommunikation
- Verarbeitungslogik
- Rechner emuliert Lift






Tests vor Ort:

- Stabilität und Fehleranfälligkeit der Kommunikation
- Verhalten des Lagerlifts



Insomnia
REST API Client

Agenda

-  Vorstellung Unternehmen und Projekt
-  Analysephase
-  Entwurfsphase
-  Implementierungsphase
-  Fazit und Ausblick

Fazit und Ausblick