

IBSYS II

Wirtschaftsinformatik [Bachelor]

Sommersemester 2022

**Softwarehandbuch**

über das Supply Chain Tool

vorgelegt bei:

**Prof. Dr.-Ing. Karl-Robert Graf**

vorgelegt von: Gruppe DR4

Gruppenmitglieder: Julius Dörenkamp, Marcel Märdian, Selim Kaan Baykara

Abgabedatum: 22.06.2022

Version: 1.0

**Inhaltsverzeichnis**

[1. Anforderungen an das Tool 3](#_Toc106612134)

[2. Aufgabe der Anwender 3](#_Toc106612135)

[3. Installation 3](#_Toc106612136)

[1. 3.1 Start aus der IDE 3](#_Toc106612137)

[2. 3.2 Start mit einzelner Datei 4](#_Toc106612138)

[3. 3.3 Start mit Heroku 4](#_Toc106612139)

[4. Frontend 4](#_Toc106612140)

[4. 4.1 Architektur 4](#_Toc106612141)

[5. 4.2 Node Pakete 5](#_Toc106612142)

[6. 4.3 Grundlagen React 5](#_Toc106612143)

[7. 4.4 Klassen 6](#_Toc106612144)

[8. 4.5 ma the() 8](#_Toc106612145)

[9. 4.6 Anleitung zur Weiterentwicklung 9](#_Toc106612146)

[5. Datenerfassung und Erstellung der Planung 9](#_Toc106612147)

[6. Backend 16](#_Toc106612148)

[7. Implementierung der Funktionalität 17](#_Toc106612149)

# Anforderungen an das Tool

Das Tool soll übersichtlich und leicht verständlich sein und zu einer Produktionsplanung und -steuerung unterstützend verwendet werden können. Grundlegend soll eine Periode eines Produktionsauftrags simuliert werden können. Daher ist eine manuelle Eingabe und Darstellung der Vertriebsplanung notwendig. Dies bedeutet, den Vertriebswunsch als auch künftige Prognosen im Tool hinterlegen zu können. Auch eine manuelle Eingabe des Direktverkaufs soll möglich sein. Des Weiteren soll eine Eingabe des Produktionsprogramms möglich sein. Die Mengenplanung mit Stücklistenauflösung für Eigenfertigprodukte und Teileverwendung für die Kaufteile soll ebenfalls im Tool abgebildet sein. Die 14 Arbeitsplätze und deren jeweilige Kapazitäten sollen auch geplant und angepasst werden können. Der Nutzer soll ebenfalls die Möglichkeit besitzen, die vorliegenden Daten vor Benutzung des Tools als XML-Datei aus vergangenen Perioden zu importieren als auch die Ergebnisse und Berechnungen des Tools im Nachgang an den erfolgreichen Durchlauf als XML-Datei zu exportieren. Um Durchlaufzeiten verbessern zu können, soll ebenfalls ein Losgrößensplitting mit Reihenfolgeplanung möglich sein. Abschließend soll auch aus internationalem Umfeld kommenden Nutzern die Benutzung des Tools ermöglicht werden. Daher ist eine Verwaltung verschiedener Fremdsprachen verfügbar.

# Aufgabe der Anwender

Die Aufgabe der Anwender ist es, einen laufenden Produktionsbetrieb mit Unterstützung des Tools für eine weitere Periode am Laufen zu halten. Hierzu dient ein einfaches Betriebsmodell, welches im Rahmen der Software abgebildet ist und Entscheidungen über Bestellaufträge, Fertigungsaufträge sowie Fertigungskapazitäten trifft. Die Anwender sollen dabei die Produktionsleistung dieses Produktionsbetriebes auf die jeweiligen Anforderungen des Marktes abstimmen und Ziele wie Durchlaufzeiten, Liefertreue, Auslastungen, Bestände, Herstellkosten und das Betriebsergebnis stets im Hinterkopf behalten.

# Installation

## 3.1 Start aus der IDE

Zur Ausführung der Software benötigt der Nutzer einen Quelltext-Editor zur Ausführung des produzierten Quellcodes. Empfohlen und ebenfalls vom Team der Studierenden genutzt wird Microsoft Visual Studio Code. Dieser kostenlose Editor ist plattformübergreifend für Windows, macOS und Linux verfügbar. Den für den Start benötigten Programmcode kann wahlweise lokal importiert werden oder über eine Verbindung zu Github, einer Software für Entwicklungsprojekte, bereitgestellt werden.

Nach erfolgreicher Einrichtung des Editors als auch der erfolgreichen Bereitstellung des Quellcodes muss der Nutzer zwei Fenster im Quelltext-Editor öffnen. Im ersten Fenster öffnet er den Quellcode mit dem Dateinamen ‘planning-service’. Dieser Teil der Software stellt das Backend dar. Das Backend wird mit dem Befehl ‘npm start run’ im Terminal gestartet. Im zweiten Fenster öffnet er den Quellcode mit dem Dateinamen ‘ibsys2erp’. Dieser Teil der Software stellt das Frontend dar. Da Frontend kann mit dem Befehl ‘npm start’ im Terminal gestartet werden. Nachdem beide Software Blöcke gestartet wurden, öffnet sich ein Browser auf dem Computer der nutzenden Person.

## 3.2 Start mit einzelner Datei

Außerdem liegen den beiden Applikationen .bat Dateien bei, mit denen es möglich ist, die Applikationen zu starten. Diese Dateien heißen beide jeweils start.bat und öffnen beim Ausführen eine Konsole, in denen die beiden Anwendungen anschließend starten und laufen. Diese Variante ist nur für Windows möglich.

Vermerk: Diese Variante funktioniert nur unter Windows.

## 3.3 Start mit Heroku

Die leichteste Variante die Applikation zu verwenden ist über den folgenden Link: <https://ibsys-dr4.herokuapp.com>

Hier muss jedoch bedacht werden, dass die Anwendung runterfährt, wenn sie lange nicht verwendet wurde. Zudem sind die Rechenleistungen, die der App hier zur Verfügung stehen, nicht sehr groß. Dementsprechend muss der Nutzer etwa 30 Sekunden bis 1 Minute nach Aufruf des Links warten, bis die Applikation gestartet ist. Danach ist der Dienst mehrere Stunden verfügbar, bevor er wieder automatisch runterfährt.

Gleiche Vorgehensweise gilt für das Backend und muss auch hier durchgeführt werden.

https://planning-service.herokuapp.com/

Vermerk: Diese Variante empfehlen wir dem Nutzer.

# Frontend

Dieses Kapitel dient dazu, einblicke in die Technik hinter dem Frontend zu gewähren und richtet sich an unsere technik-affinen Stakeholder.

## 4.1 Architektur

Das Frontend basiert auf Node.js, der gängigsten JavaScript Laufzeitumgebung. Als Paketmanager wird npm genutzt. Der Node Paket Manager ermöglicht es, sehr leicht zusätzliche Pakete zu Node.js hinzuzufügen. Dabei überprüft npm Kompatibilitäten und Abhängigkeiten. Außerdem können Updates einzelner Pakete direkt über npm installiert werden.

Als Frontend-Framework wird React eingesetzt. React ist ein komponentenbasiertes Frontend Framework. Es wird in über 10 Millionen öffentlichen Webseiten verwendet und ermöglicht es ressourcenschonende Web-Applikationen zu entwickeln.

Bei React ist jedes UI-Element eine Komponente. React ermöglicht somit, dass nur Komponenten geladen werden, welche gerade genutzt werden.

Für das Styling und das Anordnen der Komponenten wird Bootstrap verwendet. Bootstrap ist ein CSS-Frontend Framework. Es erweitert CSS um Geschaltungsvorlagen und ein Grid-System, mit denen Elemente angeordnet werden können.

## 4.2 Node Pakete

Die folgende Liste beinhaltet alle verwendeten Pakete. Zum Zeitpunkt der Fertigstellung der Entwicklung waren die unten genannten Versionen der Pakete die aktuellsten und volle Kompatibilität ist gegeben.

* @material-ui/core@4.12.4
* @tanstack/react-table@8.0.0-alpha.26
* @testing-library/jest-dom@5.16.3
* @testing-library/react@12.1.4
* @testing-library/user-event@13.5.0
* axios@0.26.1
* material-table@2.0.3
* react-dom@18.0.0
* react-languages-select@1.0.20
* react-router-dom@6.3.0
* react-scripts@5.0.0
* react-table@7.7.0
* react-toastify@8.2.0
* react@18.0.0
* reactstrap@9.0.1
* web-vitals@2.1.4

## 4.3 Grundlagen React

Als Grundlage wurde “Create React App” genutzt. Dabei handelt es sich um eine React Toolchain, welche über npm bezogen werden kann. Dieses Paket bringt alle Bausteine für eine React app mit, aber auch Scripts, welche die Entwicklung erleichtern.

“Create React App” legt dabei die Ordnerstruktur an und erstellt die ersten Dateien. Außerdem werden einige Pakete installiert, welche sehr häufig verwendet werden. Des Weiteren übernimmt das Paket die Konfiguration der build-tools.

Der grundlegende Aufbau einer React Klasse beginnt wie üblich in JavaScript mit den Imports. Darauf folgt die Initialisierung der Klasse:

export default class Eigenfertigung extends React.Component {..}

Am Anfang der Klasse lassen sich globale Variablen definieren. Nach den globalen Variablen kommt dann der Konstruktor. Dieser sieht in React meist so aus:

constructor(props) {

super(props);

..}

Dieser Konstruktor ist Voraussetzung für das Nutzen von Properties, mit denen man Daten zwischen Komponenten übergeben kann und States, mit denen man Daten innerhalb der Klasse halten kann.

In dieser Applikation werden die States sehr häufig verwendet, um die Werte des Backends zu halten. Um States zu verwenden, muss folgendes im Konstruktor eingefügt werden:

this.state = {

state1: ..;

state2: ..;

..}

Nach dem Konstruktor folgen in aller Regel alle JavaScript Funktionen, so auch in dieser Applikation.

Danach kommt ein weiterer React-Anteil der Klasse. Und zwar “render()”. Hier wird der HTML Code platziert, um das darzustellen, was der Endnutzer sehen soll. Genauer, wird der Code in den “return ()”-Teil von render() geschrieben. Zwischen render und return ist dann Platz für weitere JavaScript Funktionen, die für das UI benötigt werden.

## 4.4 Klassen

**App.js**

Die App.js Klasse ist die Hauptklasse des Frontends. Sie beinhaltet die Startseite sowie die Logik über die Kontrolle der anderen Seiten bzw. Komponenten.

Die Startseite besteht aus einem Input-Feld, um Dateien hochzuladen, einem Upload Knopf und einem Knopf, um zu der Eigenfertigungsdisposition zu gelangen.

Im Körper der Klasse App überprüfen die Funktionen checkMimeType(), maxSelectFile(), checkFileSize() die Ergebnis-XML aus der Supply Chain Simulation. In checkMimeType() wird die Dateiendung geprüft, dabei werden ausschließlich XML-Dateien zugelassen. Mit maxSelectFile() wird überprüft, dass nur eine XML-Datei auf einmal hochgeladen wird. Die Funktion checkFileSize() grenzt die Dateigröße ein. Die Eingrenzung ist ein Sicherheitsfeature, da so nicht die Möglichkeiten besteht, dass sehr große XML-Dateien hochgeladen werden, die den Server überlasten.

Die Funktion onChangeHandler()setzt den state “selectedFile” und hinterlegt die ausgewählte Datei. Das Auslösen des Knopfen “Hochladen” löst die Funktion onClickHandler() aus, welche die ausgewählte Datei an das Backend schickt. Die Funktion benachrichtigt den Nutzer außerdem über den erfolgreichen Upload und zeigt eine Progress-Leiste.

Im Bereich render() befindet sich zunächst die Logik, um zwischen den anzuzeigenden Seiten zu wechseln. Hier wird die Funktion clickBtn() verwendet, um zwischen den Switch-Cases zu wechseln. Innerhalb von render() befindet sich unter return() das, was auf der Startseite zu sehen ist. Hier kann das Styling angepasst werden und HTML-Elemente hinzugefügt werden.

**Eigenfertigungs.js**

Diese Klasse enthält die drei Tabellen für die Eigenfertigungsdisposition. Jedem Eingabe- und Anzeigefeld ist ein “State”, im Konstruktor der Klasse, zugeordnet. Nachdem die Ergebnisdatei hochgeladen wird, wertet das Backend diese aus und schickt sofort die benötigten Daten für die Eigenfertigungsdisposition. Diese Daten werden dann in die jeweiligen “States” geladen.

Wird ein Eingabefeld geändert, wird die Funktion handleChange(evt) aufgerufen, die den “State” des dazugehören Feldes aktualisiert.

Ändert man z.B. den geplanten Lagerbestand, muss die gesamte Tabelle neu berechnet werden. Um das Programm nicht zu überfordern, wird die Neuberechnung nur über die Speicherfunktion ausgeführt. Dabei errechnet die Funktion math() die Tabellen neu und speichert die Werte. Der onClickHandler2() versorgt die Kapazitätsplanung und das Backend mit den Eingaben, die der Nutzer getätigt hat.

Unter render() befindet sich der HTML Code für die drei Tabellen.

**Kapazität.js**

Die Klasse Kapazität.js hat einen ähnlichen Aufbau wie die Klasse Eigenfertigung.js. Für jedes Feld gibt es einen “State” und für die Aktualisierung die Funktion handleChange(evt). Die Werte werden erneut aus der Antwort des Backends bezogen, die auf das Abschicken der Eigendisposition folgt.

**Prognose.js**

In der Klasse wird der Vertriebswunsch und die Prognose der nächsten Personen über eine Durchschnittsberechnung glattgezogen. Dafür werden in render() zwei Tabellen gezeichnet. Die erste fordert den Nutzer auf, die Prognose aus dem SCS einzugeben, der Vertriebwunsch ist bereits eingetragen. Klickt der Nutzer dann auf berechnen, wird aus dem Vertriebsplan ein Produktionsplan erstellt.

**Splitting.js**

Diese Klasse wird aufgerufen, nachdem die Eigendisposition abgeschlossen ist. In einer Übersicht werden dem Nutzer die Produktionsmengen angezeigt und über Eingabefelder kann der Auftrag gesplittet werden unter der Angabe, wann der Auftrag ausgeführt werden soll (Position). Die Eingaben werden erneut über die “states” der Klasse ausgewertet. Ist der Nutzer mit allen Eingaben fertig und klickt auf Senden, so werden die States an das Backend übergeben.

**Kaufdisposition.js**

In der Kaufposition sind die Stammdaten hinterlegt, damit der Nutzer Anpassung an die vom Tool vorgeschlagenen Bestellungen machen kann, ohne das Handbuch aufschlagen zu müssen.

Der Rest der Klasse folgt dem Stil der Eigenfertigungsdisposition.

**Direktverkauf.js**

Die Klasse Direktverkauf ist die letzte Seite des der Applikation. Auch hier ist jedem Eingabefeld ein State zugeordnet. Mit onClickHandler() werdem dem Backend dabei die Daten übergeben. Ausgelöst wird die Funktion durch das Betätigen des Knopfes “Speichern und Senden”. Darunter befindet sich der Knopf “XML-Datei herunterladen”. Dieser Knopf setzt die Get-Request über die Funktion request() an das Backend ab, welches mit der Eingabe-XML antwortet.

**CSS**

Zu jeder JavaScript Klasse gibt es außerdem eine CSS-Datei, welche das Styling des HTML-Anteil der JavaScript Klassen anpasst.

## 4.5 ma th()

Die im vorherigen Abschnitt angesprochene Funktion “math()” ist die einzige Anwendungslogik im Frontend. Alles andere ist in das Backend ausgelagert. Diese Entscheidung wurde getroffen, damit nicht für jede geänderte Zahl das Backend angefragt werden muss. Diese Maßnahme ist ein relativ leichter Schutz vor Überlastung des Backends. Zudem werden so je nach System längere Wartezeiten umgangen.

Aufgrund der Komplexität wird die mathe() Funktion im Folgenden erläutert.

Zunächst wird die Funktion mit async deklariert, somit können die Rechenoperationen mit “await” deklariert werden. Dadurch wird erst in die nächste Operation gewechselt, wenn die vorherige fertig ist. Dieser Aufbau ist nötig, da die einzelnen Rechnungen aufeinander aufbauen. Das Ergebnis der einen Zeile wird in der nächsten verwendet, würde die Funktion also zu früh zur nächsten Operation gehen, würden die benötigten Ergebnisse fehlen.

Zu Beginn werden erst alle benötigten Werte der “States” aus den Input Feldern zurück in die persistenten Daten geschrieben, die aus dem Backend gekommen sind.

Anschließend werden die Zeilen schrittweise neu berechnet. Die Artikel sind hierarchisch angeordnet sind, produziert man eine gewisse Menge von Artikel X, so benötigt man dafür wiederum eine bestimmte Menge von Artikel Y usw. Daher werden die Zeilen entsprechend ihrer Hierarchie neu berechnet. So hat die Produktionsmenge von Artikel 51 Einfluss auf Artikel 18,19,25 und 31. Also berechnet die Funktion im ersten Schritt den Artikel 51 neu und mit dem Ergebnis werden die abhängigen Artikel berechnet. Im Kern macht die Rechnung Folgendes:

**Vertriebswunsch/Benötigte Menge + Geplanter Lagerbestand - Aktueller Lagerbestand - Warteliste - In Bearbeitung = Produktionsmenge**

Das Ergebnis der Funktion ist, dass die Daten für das Backend mit den vom User angepassten Werten upgedatet sind und dass die korrekten Produktionszahlen in den Tabellen stehen.

Würde man jetzt die Daten zurückschicken, sind diese akkurat und aktualisiert.

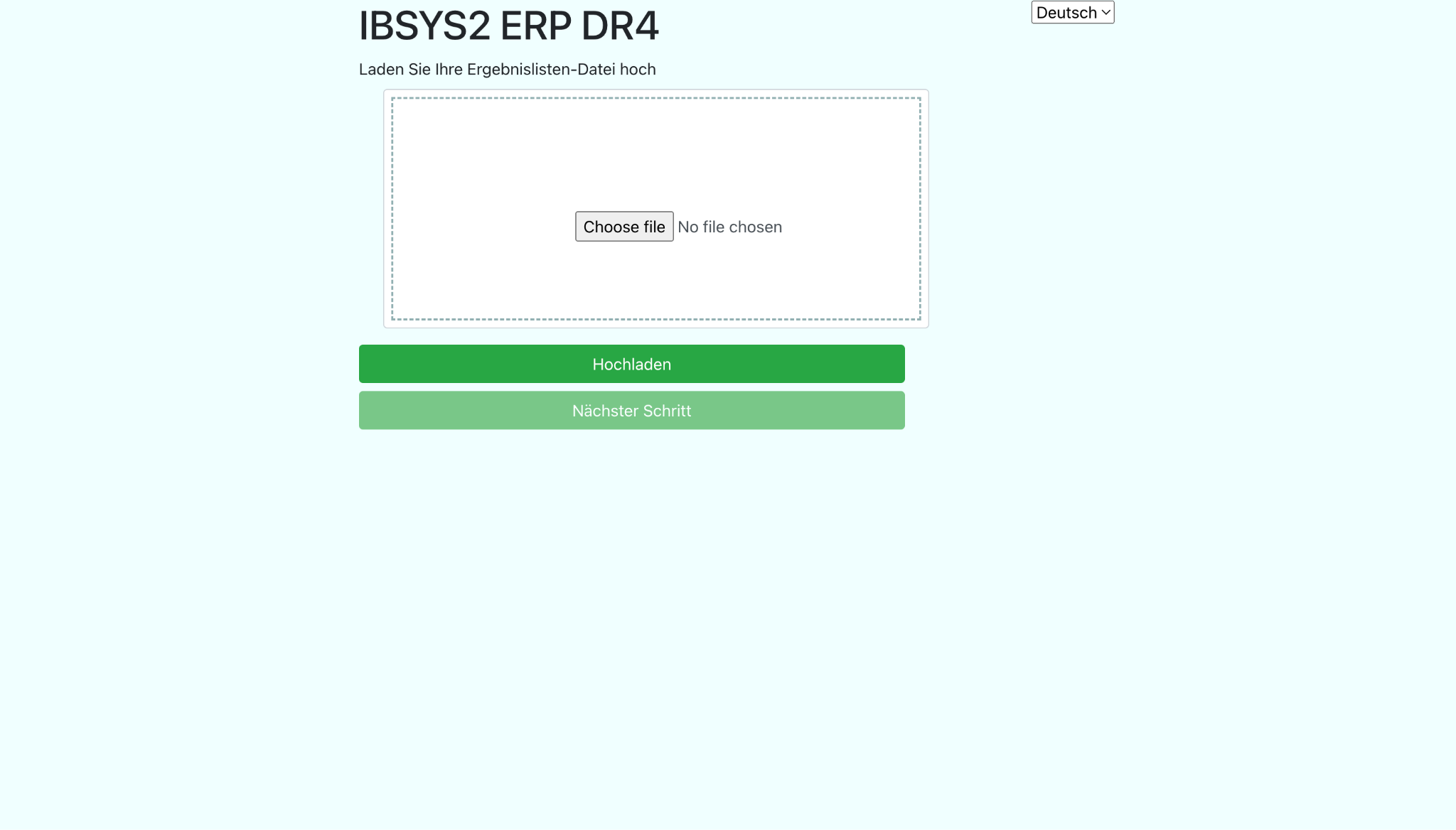
## 4.6 Anleitung zur Weiterentwicklung

Für die Entwicklung wird Visual Studio Code empfohlen.

Über die interne Konsole kann mit dem Skript “npm start” das Frontend gestartet werden. Änderungen an den Dateien werden nach dem Speichern sofort angezeigt, ohne dass eine Aktualisierung der Seite oder ein Neustart des Service nötig ist.

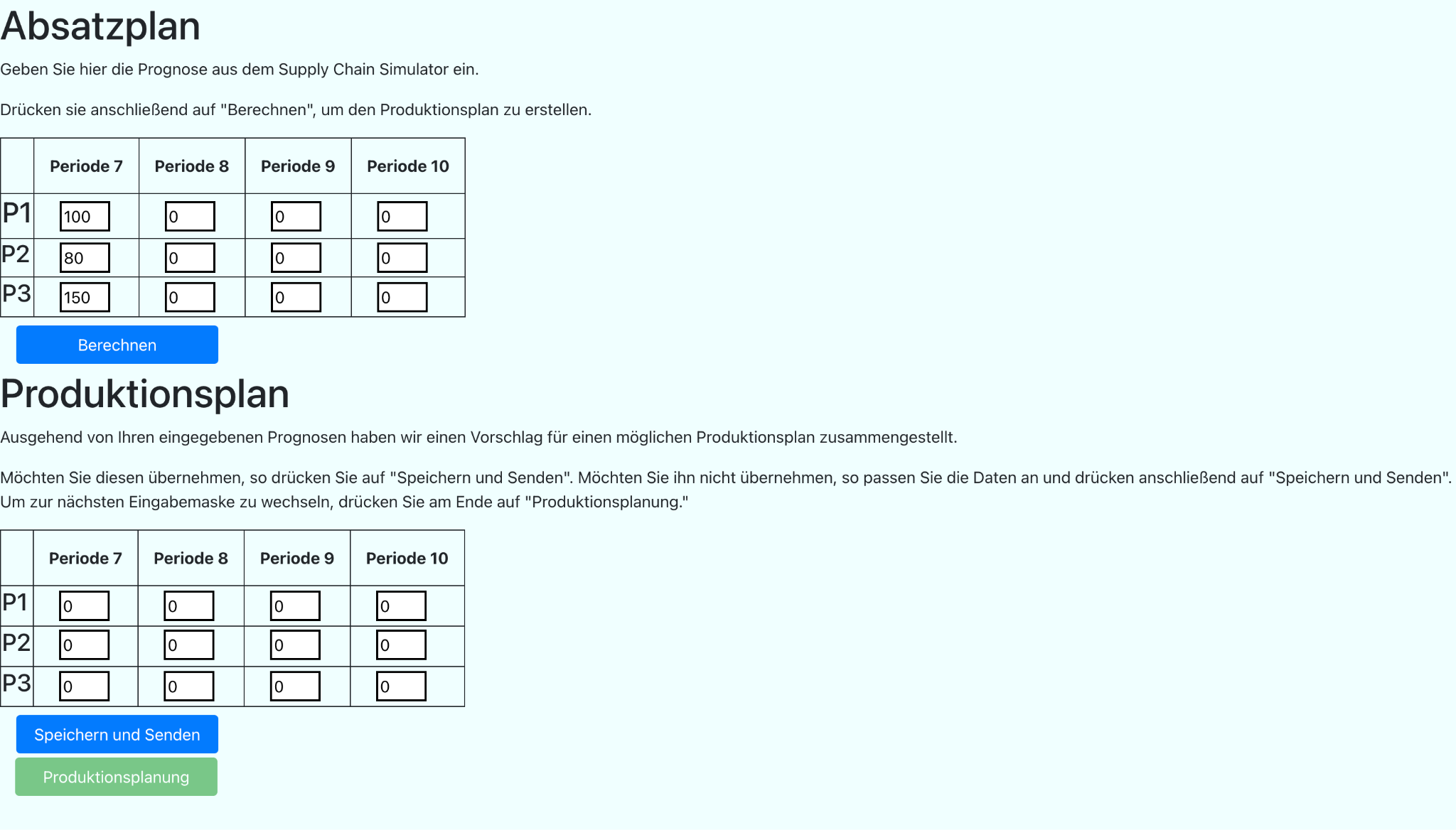
# Datenerfassung und Erstellung der Planung

Startseite



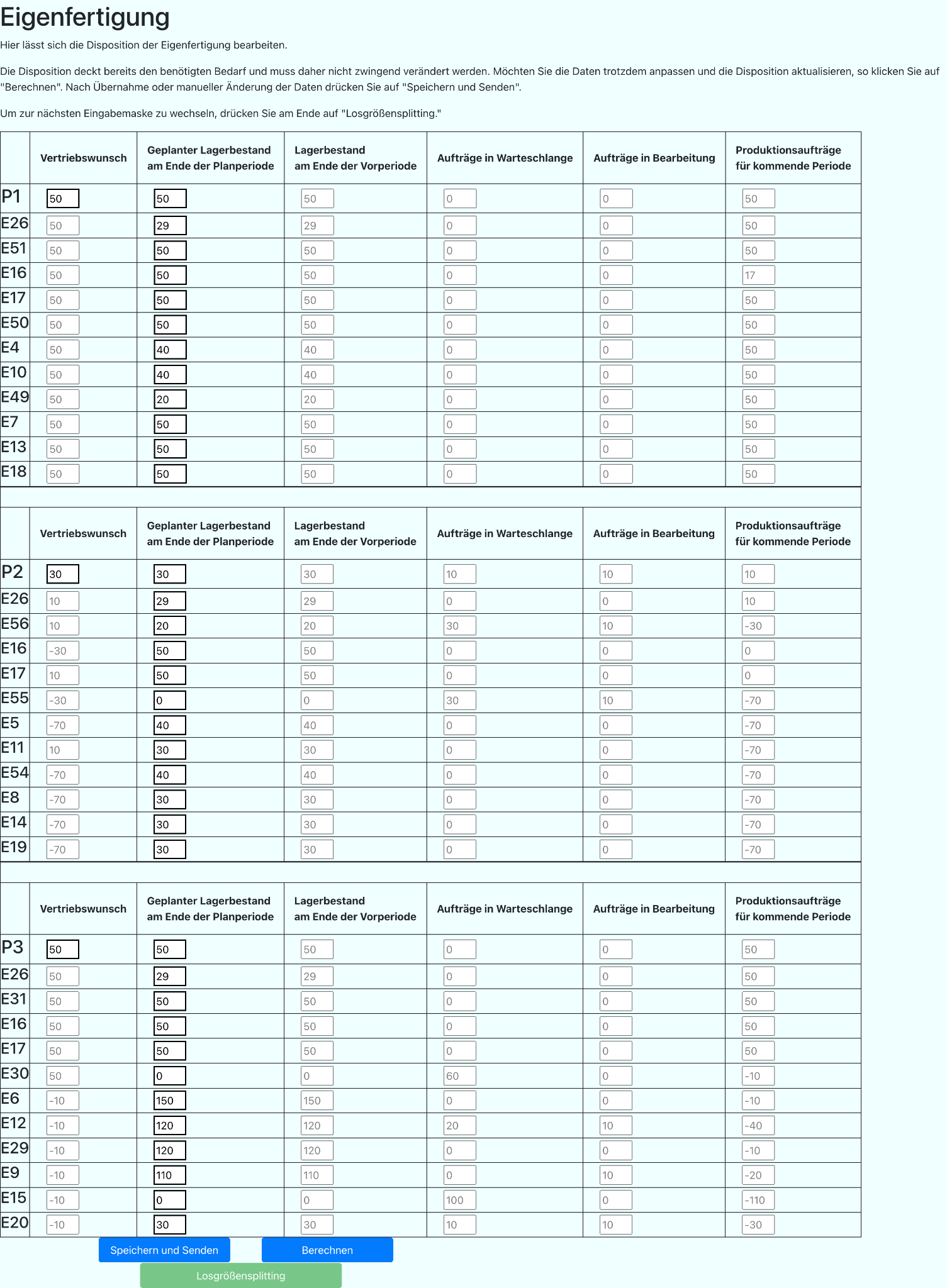
Grundlegend verfügt die Software über die Möglichkeit, die Oberfläche in Zukunft in verschiedenen Sprachen anzeigen zu lassen. Hierzu muss der Nutzer im oberen Bereich das *Dropdown Menü* öffnen und kann zwischen den Sprachen deutsch und englisch wählen. Über den Button *Choose file* im unteren Bereich kann der Nutzer das XML File mit den Ergebnisdaten einer vergangenen Periode auswählen und anschließend über den Button *Hochladen* hochladen. Während des Ladevorgangs wird im noch grauen Balken der Status des Uploads angezeigt. Sobald das XML erfolgreich hochgeladen wurde, erscheint im rechten oberen Eck eine Meldung mit der Nachricht *Upload successfull*. Anschließend kann über den Button *Nächster Schritt* zur nächsten Eingabemaske gewechselt werden.

Absatzplan und Produktionsplan



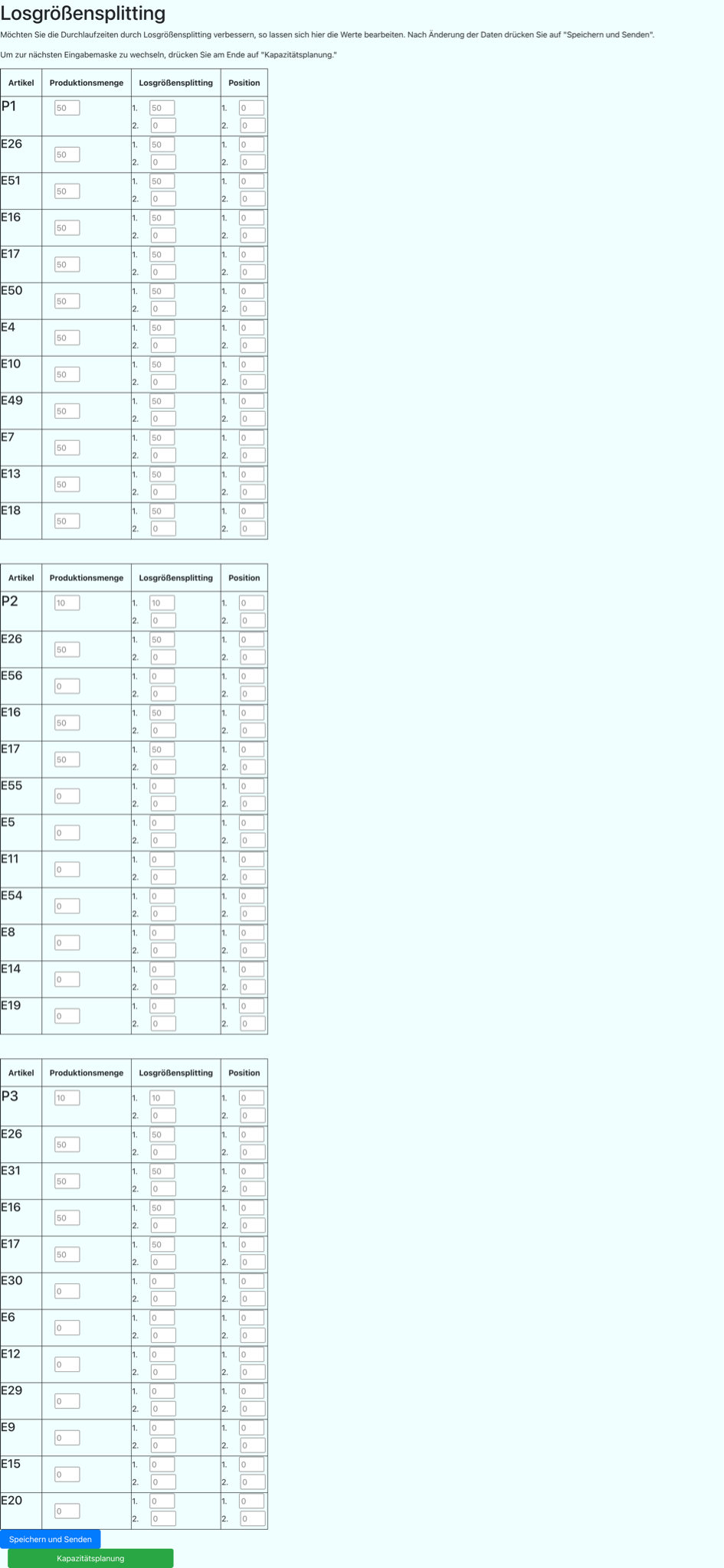
Nach jedem Simulationsdurchlauf werden für die kommenden drei weiteren Perioden Prognosen für die Enderzeugnisproduke P1, P2 und P3 bereitgestellt. Um im nächsten Schritt beim Planen und Kalkulieren der Teilebestellung diese Prognosen berücksichtigen zu können, sollten diese an dieser Stelle in der Software eingepflegt werden. Über den Button *Berechnen* erhält der Nutzer einen Vorschlag über einen möglichen Produktionsplan. Diesen kann er annehmen oder ablehnen und Änderungen vornehmen. Über den Button *Speichern und Senden* speichert der Nutzer am Ende die Daten. Beim Klicken auf den Button *Produktionsplanung* gelangt der Nutzer zur nächsten Eingabemaske.

Eigenfertigung



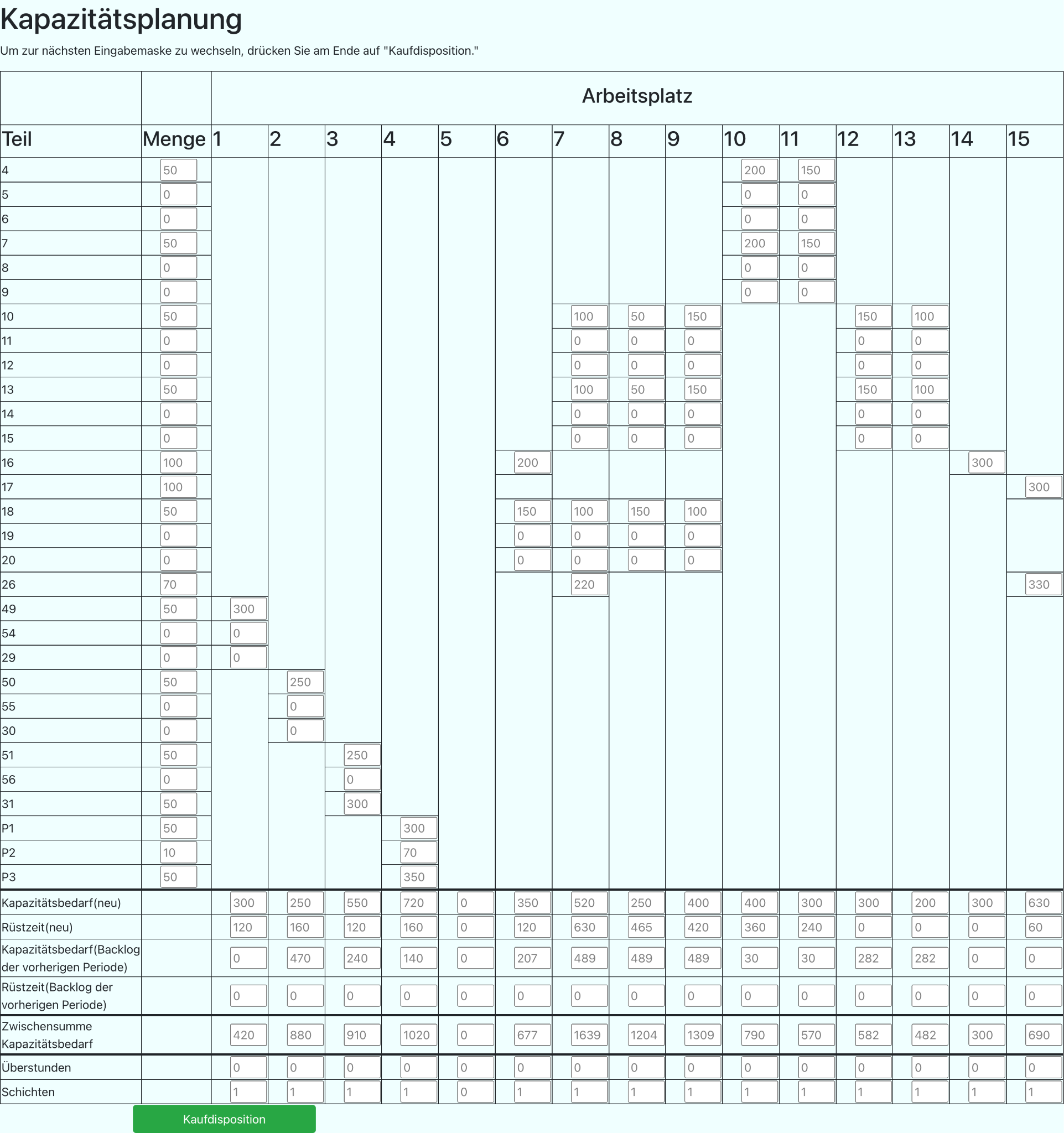
In der nächsten Eingabemaske erhält der Nutzer eine Übersicht der Eigenfertigung. Grundlegend können hier alle Zellen bearbeitet werden, welche einen schwarzen und dicken Rand aufweisen. Die restlichen Zellen werden automatisch mit Werten befüllt und können daher nicht verändert werden. Visuell unterteilt wird die Eigenfertigung in den Abschnitt P1 (Kinderfahrrad), P2 (Damenfahrrad) und P3 (Herrenfahrrad). Im Normalfall reich für die Disposition der Eigenfertigungsprodukte eine Betrachtung der nächsten Periode aus. Abhängig vom Ziel des Nutzers kann ebenfalls der geplante Lagerbestand am Ende der aktuellen Planperiode verändert werden, um den Lagerbestand beispielsweise zu reduzieren, um Kapitalbindung zu vermeiden. Über den Button *Berechnen* erhält der Nutzer eine Neuberechnung der eingegebenen Daten. Über den Button *Speichern und Senden* speichert der Nutzer die eingegebenen Daten. Sind all diese Daten eingegeben und gespeichert, so kann der Nutzer im unteren Bereich über den Button *Losgrößensplitting* zur nächsten Eingabemaske wechseln.

Losgrößensplitting



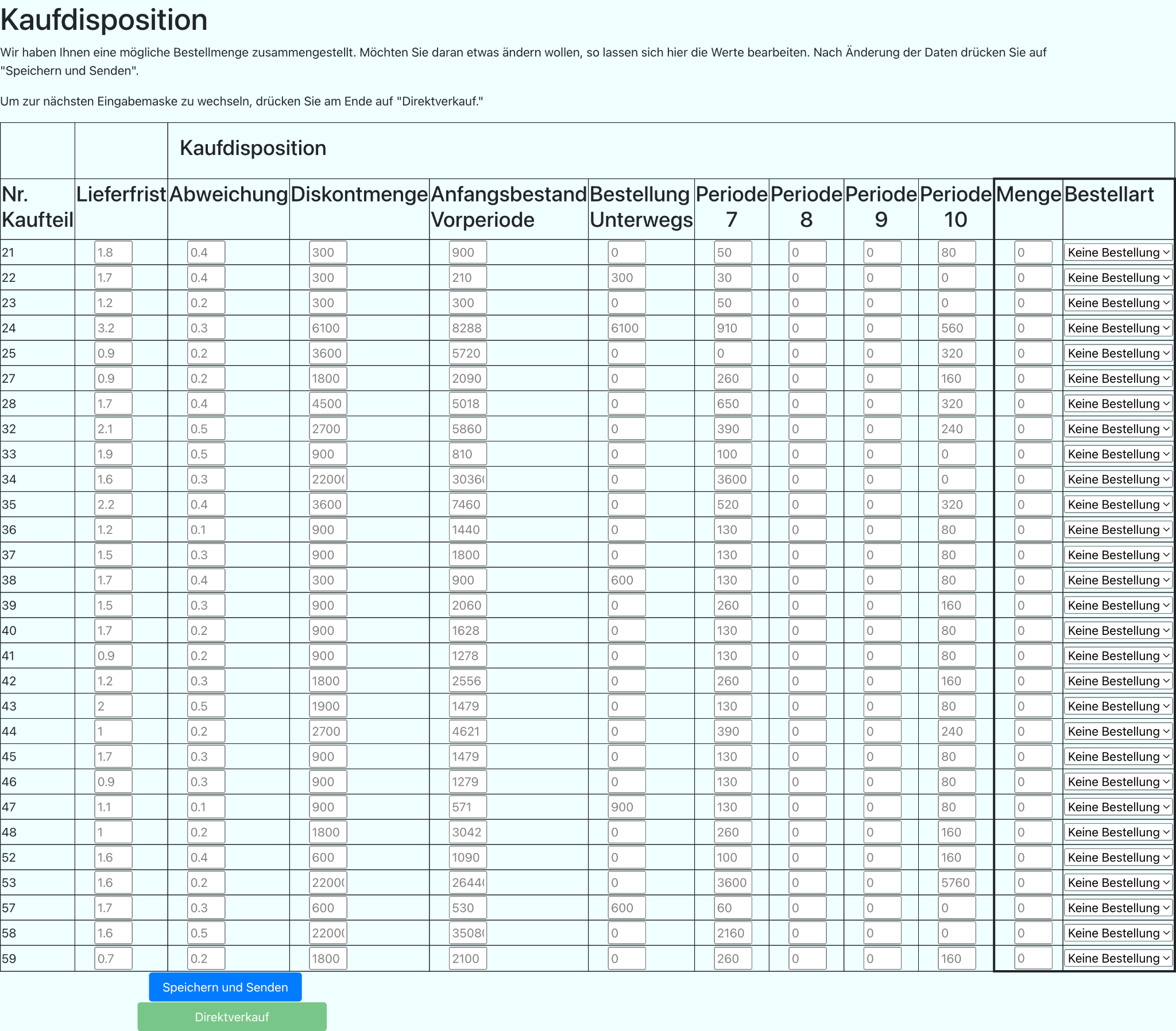
In der nächsten Eingabemaske erhält der Nutzer eine Übersicht des Losgrößensplittings. Grundlegend können hier alle Zellen bearbeitet werden und die Aufträge gesplittet werden, um beispielsweise Durchlaufzeiten zu optimieren. Über den Button *Speichern und Senden* speichert der Nutzer die eingegebenen Daten. Sind all diese Daten eingegeben und gespeichert, so kann der Nutzer im unteren Bereich über den Button *Kapazitätsplanung* zur nächsten Eingabemaske wechseln.

Kapazitätsplanung



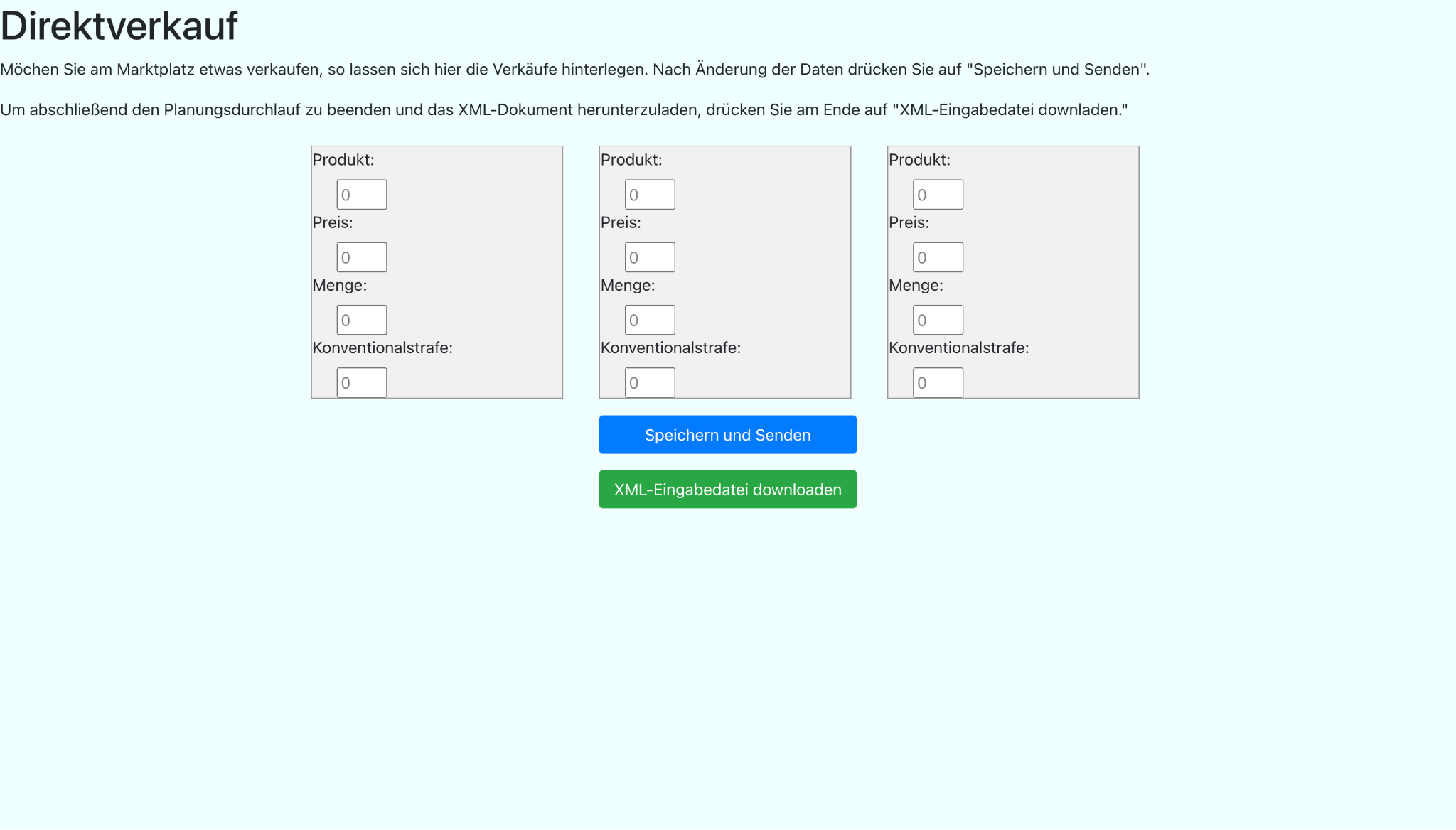
In der nächsten Eingabemaske erhält der Nutzer eine Information über die ermittelten Fertigungsaufträge. Darüber lässt sich für jeden der 14 Arbeitsplätze ein Kapazitätsbedarf ableiten. Der neue Kapazitätsbedarf je Arbeitsplatz und Periode ergibt sich aus der Summe der Fertigungs- und Rüstzeiten derjenigen Aufträge, die in dieser Periode an diesem Arbeitsplatz bearbeitet werden sollen. Sollte die dadurch entstehende Gesamtkapazität größer sein als die normale Grundkapazität, so müssen wir eine Kapazitätserweiterung durchführen und Überstunden und Schichtarbeit einführen. Diese beiden Fälle sind in den untersten beiden Zeilen abgebildet. Über den Button *Kaufdisposition* kann der Nutzer zur nächsten Eingabemaske wechseln.

Kaufdisposition



Abhängig von Prognosen und Lagerbeständen muss der Nutzer Kaufteile nachbestellen. Dabei können Eilaufträge mit konstanter Lieferzeit oder Normalaufträge mit variabler Lieferzeit in Auftrag gegeben werden. Dies lässt sich über das Dropdown-Menü in der letzten Spalte vom Nutzer festlegen. Wurden alle Werte eingetragen und die Bestellarten definiert, so gelangt der Nutzer über Button *Direktverkauf* zur nächsten Eingabemaske.

Direktverkauf



Über einen weiteren Eingabeblock kann, zusätzlich zum Vertriebswunsch, der Verkauf von Endprodukten ab Lager (Direktverkauf) veranlasst werden. Bei Direktverkäufen kann dabei abweichend von vorheriger Preisregulatorik der Verkaufspreis beliebig vorgegeben werden. Über den Button *Speichern und Senden* speichert der Nutzer die eingegebenen Daten. Abschließend kann über den Button *XML-Eingabedatei downloaden* die Berechnung und ein Vorschlag für die nachfolgende Periode als XML-Datei heruntergeladen werden.

# Backend

Das Backend ist eine Serveranwendung mit einer REST-Schnittstelle. Innerhalb des Backends liegt die Logik der Anwendung. Das heißt alles was für die Produktionsplanung anwendungsseitig berechnet wird, wird tatsächlich im Backend berechnet. Das Backend beherbergt somit die Umsetzung der Funktionalität der Anwendung. Abgerufen wird diese Funktionalität seitens Frontend jeweils durch HTTP-Requests. Dieses Kapitel dient dazu, Einblicke in die Technik hinter dem Backend zu gewähren und richtet sich an unsere technik-affinen Stakeholder.

Das Backend wurde in der Programmiersprache TypeScript entwickelt. TypeScript ergänzt die altbekannte Programmiersprache JavaScript um die Möglichkeit der Typisierung. Dadurch werden altbekannte JavaScript-Fehler, wie der Zugriff auf eine Property, welche in einem Objekt gar nicht existiert, vermieden. TypeScript-Code wird vor der Ausführung durch den TypeScript-Compiler in JavaScript-Code umgewandelt. Editoren wie Visual-Studio-Code können den Type-Script-Code während der Entwicklung kompilieren, wodurch o.g. Fehler schon vor der Laufzeit erkannt werden können.

Die Laufzeitumgebung für das Backend ist Node.js. JavaScript war ursprünglich für die Ausführung innerhalb eines Webbrowsers gedacht. Node.js ermöglicht es, JavaScript-Code außerhalb des Webbrowsers auszuführen und ermöglicht somit die Implementierung von Serveranwendungen in JavaScript bzw. TypeScript. Der Vorteil dabei ist, dass Node.js ressourcensparend ist und eine gute Performance verspricht.

Die Serveranwendung wurde mit dem Framework NestJS implementiert. Die Verwendung von NestJS hat den Vorteil, dass der Entwickler sowohl beim Entwickeln als auch beim Warten der Software sich lediglich um die fachliche Logik kümmern muss, die technische Logik wird größtenteils vom Framework übernommen. Dies ermöglicht ein effizienteres Arbeiten und eine bessere Fokussierung auf die Wünsche des Kunden.

Als Datenbank verwendet das Backend eine SQLite-Datenbank. Da in der Produktionsplanung viele unterschiedliche Geschäftsobjekte, mit komplexen Beziehungen zueinander verwendet werden, haben wir uns für eine relationale Datenbank entschieden.

# Implementierung der Funktionalität

In diesem Kapitel wird die interne Funktionsweise der Software skizziert. Dabei wird auf die einzelnen Funktionen (Einpflegen der XML, Produktionsprogramm, Eigenfertigungsdisposition, Kapazitätsplanung, Kaufdisposition, Direktverkauf, Generierung der Ergebnis-XML) eingegangen. Für jede Funktion wird jeweils der Datenverarbeitungsprozess dargestellt. Dieses Kapitel dient dazu, den gesamten Planungsprozess, der durch unsere Software realisiert wird, nachvollziehbar für den Anwender zu machen.

Einpflegen der XML

Wie in der Anleitung dargestellt, pflegt der Anwender eine XML mit den relevanten Daten, wie den Lagerbeständen, ausstehenden Bestellungen etc. für die gewünschte Planungsperiode ein. Mithilfe dieser Daten wird zunächst der Datenbestand in der Datenbank aktualisiert. Die Bestände der Artikel werden ebenfalls aktualisiert. Aufträge, welche noch in Bearbeitung sind, werden eingepflegt, Auftragswartelisten werden aktualisiert, die Verbindlichen Aufträge für die Aktuelle Periode werden eingespeichert und bestellte Kaufteile, welche zukünftig eintreffen werden, werden ebenfalls eingepflegt.

Produktionsprogramm

Wie bereits erwähnt werden die verbindlichen Aufträge für die aktuelle Periode aus der XML entnommen. Um ein Produktionsprogramm, welches die drei nachfolgenden Perioden ebenfalls mitberücksichtigt, zu erstellen, gibt der Benutzer in der Maske die Vertriebsprognosen für die kommenden 3 Perioden ein. Nach dem Bestätigen der Eingabe berechnet die Software einen Planungsvorschlag, in den die Kapazitäten möglichst gleichmäßig über die Perioden verteilt werden. Der Benutzer hat noch die Möglichkeit, den Vorschlag anzupassen. Nach der Bestätigung wird das neue Produktionsprogramm in der Datenbank eingepflegt.

Eigenfertigungsdisposition

Diese Funktion erspart dem Anwender eine Vielzahl an Arbeitsstunden. Ausgehend vom Produktionsprogramm für die aktuelle Periode wird hier die Anzahl der zu produzierenden Eigenfertigungsteile in der aktuellen Periode bestimmt. Sobald der Benutzer das Produktionsprogramm bestätigt hat, wird der Berechnungsprozess für die Eigenfertigungsdisposition angestoßen. In der Datenbank ist für jeden Eigenfertigungsartikel hinterlegt, welche anderen Eigenfertigungsartikel direkt für dessen Herstellung benötigt werden. Für die Artikel P1, P2 und P3 wird jeweils nacheinander die Disposition berechnet. Das Produktionsziel für einen P-Artikel wird folgendermaßen berechnet: Der Vertriebswunsch und der gewünschte Lagerbestand nach der aktuellen Periode werden zusammenaddiert. Standardmäßig rechnet das System mit einem gezielten Bestand, welches dem aktuellen Lagerbestand entspricht. Davon wird die Stückzahl der Aufträge, die bereits in Bearbeitung sind abgezogen. Zusätzlich wird auch die Stückzahl der Aufträge, die sich derzeit in der Warteschlange befinden abgezogen. Das Ergebnis stellt das verbindliche Produktionsziel dar. Auf Basis des Produktionsziels wird nun das Produktionsziel für die direkt untergeordneten Artikel berechnet. Für P1 beispielsweise sind das die Artikel E26 und E51. Als Produktionsziel des übergeordneten Artikels liefert dabei den Veräußerungswunsch. Sollen z.B. 100 P1 produziert werden, ist der Veräußerungswunsch für E51 und E26 jeweils 100. Wie bei den Fahrrädern, wird dazu wieder der Lagerbestand addiert, die Stückzahl der Aufträge in Bearbeitung wird addiert, die Stückzahl der Aufträge in der Warteschlange wird dazu addiert und daraus ergibt sich das Produktionsziel. Auf Basis dieses Produktionsziel werden wiederum die Dispositionen der untergeordneten Artikel berechnet, bis die Disposition für alle Eigenfertigungsteile komplett ist. Dieser generierte Dispositionsvorschlag wird dem Anwender auf der Benutzeroberfläche angezeigt, woraufhin der Anwender den gewünschten Lagerbestand nach der Periode anpassen kann und sich die Disposition automatisch daraufhin anpasst.

Losgrößensplitting

Hier kann der Benutzer die Reihenfolge der Produzierten Lose beeinflussen und zusätzlich die Produktionslose in Teillose aufteilen. Dies wird anschließend in der Ergebnis-XML-Datei berücksichtigt.

Kapazitätsplanung

Sobald das Losgrößensplitting vom Benutzer bestätigt wurde, beginnt im System die Berechnung der Kapazitätsplanung. Die Planung erfolgt Arbeitsplatz-weise. Für jeden Arbeitsplatz, wird für jedes einzelne Produkt, welches am jeweiligen Arbeitsplatz produziert wird, die gesamte benötigte Arbeitszeit an dem jeweiligen Arbeitsplatz berechnet. Die Information, welches Produkt, welchen Arbeitsplatz durchläuft und wie viel Rüstzeit und wie viel Produktionszeit pro Stück benötigt werden, wird aus den Stammdaten in der Datenbank entnommen. Die Menge der zu produzierenden Artikel wird aus der Eigenfertigungsdisposition entnommen. Wurden alle Produktionsvorgänge für einen Arbeitsplatz ermittelt, werden als nächstes die Arbeitszeiten für Artikel welche in der Warteschlange sind oder noch in Bearbeitung sind berechnet. Schließlich wird der Gesamtkapazitätsbedarf des jeweiligen Arbeitsplatzes, inklusive Rüstzeiten ausgerechnet. Daraus wird ermittelt, wie viele Überstunden benötigt werden und wie viel Schichten gearbeitet werden muss. Alle ermittelten Informationen sind anschließend für den Benutzer sichtbar.

Kaufdisposition

Bestätigt der Benutzer die Kapazitätsplanung, beginnt die Berechnung der Kaufdisposition. Diese Erfolgt schrittweise für jedes einzelne Kaufteil. In den Stammdaten der Datenbank ist für jeden Artikel hinterlegt, wie lange die Lieferzeit im Normalfall ist, wie hoch die Abweichungen von dieser Lieferzeit ausfallen können, bei welcher Bestellmenge es einen Diskont vom Lieferanten gibt und wie viele dieser Teile in jeweils ein P1-, ein P2- und ein P3-Fahrrad einfließt. Aus dem Produktionsprogramm wird entnommen, wie viele Fahrräder jeweils in der aktuellen und in den kommenden Perioden hergestellt werden sollen. Aus diesen gegebenen Informationen wird für jeden Artikel ermittelt, wie hoch der Gesamtbedarf in dieser und in den kommenden Perioden voraussichtlich sein wird. Des Weiteren wird aus der Datenbank entnommen, ob Bestellungen, welche in der Zukunft eintreffen werden, vorhanden sind. Zunächst wird überprüft ob der aktuelle Bestand des Artikels für alle vier Perioden ausreicht. Ist das der Fall, wird keine Bestellung für diesen Artikel angelegt. Ist das nicht der Fall, wird überprüft, ob der Artikelbestand für die aktuelle Periode und für die kommenden 2 Perioden ausreicht. Ist das der Fall wird folgendermaßen berechnet für wie viele Perioden genau der Bestand noch ausreicht:

Zeitliche Verfügbarkeit =

3 + ( (Bestand - [Bedarf der 3 Perioden]) / [Bedarf der letzten Periode] )

Der Summand 3 ergibt sich daraus, dass der Bestand für die kompletten 3 Perioden ausreicht. Anschließend wird der Restbestand, der nach den 3 Perioden übrigbleibt (Bestand – [Bedarf der 3 Perioden]) mit dem Bedarf der letzten Periode ins Verhältnis gesetzt. Unter der Annahme, dass die Artikel täglich in gleichmäßigen Mengen verbraucht werden, gibt dieses Verhältnis an, zu welcher Zeit innerhalb der Periode der Bestand aufgebraucht sein wird. Beträgt der Restbestand z.B. 250 und der Bedarf der letzten Periode 500, ergibt sich das Verhältnis 0,5. Das bedeutet zur Mitte der Periode wird der Bestand aufgebraucht. Insgesamt reicht somit der Bestand für 3,5 Perioden aus.

Sofern eine offene Bestellung für diesen Artikel vorliegt, wird die berechnete zeitliche Verfügbarkeit mit der ausstehenden Lieferdauer der Bestellung verglichen. Diese Ergibt sich aus der maximalen Lieferzeit des Artikels (Lieferzeit + Abweichung). Wurde die Bestellung eine Periode zuvor betätigt, wird der Betrag 1 abgezogen, wurde sie zwei Perioden zuvor getätigt, wird der Betrag 2 abgezogen usw. Ist die ermittelte ausstehende Lieferzeit kleiner als die zeitliche Verfügbarkeit des Lagerbestandes, bedeutet dies, dass der Nachschub ankommt, bevor der Lagerbestand aufgebraucht ist. Somit wird die zeitliche Verfügbarkeit folgendermaßen Korrigiert:

Zeitliche Verfügbarkeit =

3 + ( (Bestand **+ [Eintreffende Bestellmenge]** - [Bedarf der 3 Perioden]) / [Bedarf der letzten Periode] )

Die zeitliche Verfügbarkeit wird anschließend mit der maximalen Lieferzeit verglichen. Ist die Verfügbarkeit größer als die maximale Lieferzeit, und liegt zwischen der zeitlichen Verfügbarkeit und der maximalen Lieferzeit mindestens eine ganze Periode, muss keine Bestellung getätigt werden. Der Puffer von einer Periode ist wichtig, um zukünftige Eilbestellungen zu vermeiden.

Ist die zeitliche Verfügbarkeit zwar größer als die maximale Lieferzeit, aber die Differenz kleiner als 1, wird eine Normalbestellung mit der Diskontmenge als Bestellmenge aufgegeben. Ist die zeitliche Verfügbarkeit kleiner als die maximale Lieferzeit, muss eine Eilbestellung aufgegeben werden.

Der erläuterte Algorithmus wird ausgeführt, sofern der Lagerbestand für die 3 Perioden ausreicht. Ist dies nicht der Fall und der Lagerbestand reicht für 2 ganze Perioden aus, wird derselbe Algorithmus ausgehend von 2 Perioden durchgeführt. Anstatt der letzten Periode, wird hier der Bedarf der dritten Periode betrachtet, da der Lagerbestand in dieser Periode aufgebraucht wird. Die zeitliche Verfügbarkeit berechnet sich nun folgendermaßen:

Zeitliche Verfügbarkeit =

**2** + ( (Bestand **+ [Eintreffende Bestellmenge]** - [Bedarf der 3 Perioden]) / [Bedarf der dritten Periode] )

Ansonsten erfolgt alles wie zuvor beschrieben. Reicht der Bestand nicht für 2 ganze Perioden aus, sondern nur für eine, erfolgt dieselbe Berechnung, jedoch steht anstelle der 2 eine 1 in der Formel und die Restmenge wird mit dem Bedarf der zweiten Periode ins Verhältnis gesetzt. Reicht der Bestand auch nicht für eine ganze Periode aus, muss in jedem Fall eine Eilbestellung durchgeführt werden.

Dieser Algorithmus wird für jeden einzelnen Kaufartikel ausgeführt. Anschließend wird dem Benutzer für jeden Artikel angezeigt, ob eine Bestellung durchgeführt werden muss, welche Art von Bestellung durchgeführt werden muss und wie hoch die Bestellmenge ist. Die Bestellmenge entspricht standardmäßig immer der Diskontmenge. Der Benutzer kann diese jedoch anpassen.

Direktverkauf

Die Software bietet zudem an einen, Direktverkäufe zu planen. Dazu wird können 3 Direktverkaufpositionen angelegt werden, welche in der Ergebnis-XML berücksichtigt werden.

Ergebnis-XML

Nach der Planung der Periode besteht die Möglichkeit für den Benutzer die Ergebnis-XML für die jeweilige Periode zu generieren. Dieser Enthält die folgenden wichtigen Knoten:

* sellwish
* selldirect
* orderlist
* productionlist
* workingtimelist

Sellwish enthält jeweils den aktuellen Vertriebswunsch für alle 3 Fahrräder. Selldirect Enthält Informationen zu eventuellen Direktverkäufen in der aktuellen Periode, inklusive der Verkausfsmenge, dem Preis und der Konventionalstrafe. Orderlist enthält Informationen zu aktuellen Bestellungen, inklusive der Menge und dem Bestellmodus. Productionlist enthält die Produktionslose aller Eigenfertigungsteile in der korrekten Produktionsreihenfolge mit der jeweiligen Losgröße. Die Workingtimelist enthält die Schichten und Überstunden aller Arbeitsplätze. Die Ergebnis-XML dient zur Weiterverarbeitung der Planungsdaten in anderen Systemen.