Unihockeyball Bestell-App

Studienarbeit

Studiengang Informatik
OST - Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Herbstsemester 2021

Autoren:	Simon Hager
	Philipp Emmenegger
	Joel Schaltegger
Version:	1.0
Datum:	24. Dezember 2021
Betreuer:	Lukas Kretschmar
Projektpartner:	Techpark Rapperswil

Abstract

Ausgangslage

Im Techpark der OST Rapperswil wird eine Produktionszelle für das vollautomatisierte Herstellen von benutzerdefinierten Unihockeybällen als Show-Case betrieben. Ball-Bestellungen können aktuell nur über ein fixiertes Display an der Produktionszelle mit Direktanbindung an die Maschinensteuerung der Produktionszelle getätigt werden. Zur benutzerdefinierten Abholung wird eine Quittung mit einem QR-Code herausgegeben.

In Vorarbeiten vom Frühlingssemester 2021 wurden die Anforderungen für die Bestell-App definiert sowie die SAP Schnittstellen evaluiert. Diese Arbeiten sind Ausgangslage für diese Semesterarbeit.

Ziel der Arbeit

Innerhalb dieser Semesterarbeit soll eine funktionierende Bestell-App entwickelt werden, worüber Ball-Bestellungen an die Produktionszelle getätigt werden können.

Dazu soll eine Anbindung an die SAP Instanz der Smart Factory realisiert werden. Die SAP Instanz soll als neue Schnittstelle zwischen Bestell-App und Produktionszelle dienen. Darüber werden die Bestellungen verwaltet und Anfragen an die Maschinensteuerung weitergeleitet. Für den Aufbau dieser Systemkomponente sind Wirtschaftsinformatik-Studierende der OST St. Gallen zuständig. Weiter soll in der Bestell-App sichtbar werden, sobald eine Bestellung abholbereit ist. Der anschliessende Abholprozess, insbesondere die Herausgabe von Bällen aus dem Zwischenlager, ist ebenfalls in die Bestell-App integriert. Zusätzlich sollen Produktionsdaten aus der MindSphere Cloud heruntergeladen und in der Bestell-App visualisiert werden können.

Dafür sind eine geeignete Systemarchitektur für die Bestell-App und verwendende Schnittstellen zum SAP zu definieren. Darauf aufbauend soll die Bestell-App realisiert und die Kommunikation zum SAP sowie zu MindSphere sichergestellt werden. Ausreichende Code-Qualität wird durch Unit-Testing gewährleistet. Basierend auf der evaluierten Deploymentstrategie wird ein Software-Release umgesetzt.

Sekundär soll die Entwicklung mit allen Stakeholdern koordiniert werden. Das beinhaltet eine regelmässige Abstimmung mit den Studierenden der OST St. Gallen. In der Evaluation des geeignesten Tech-Stacks soll Siemens Mendix berücksichtigt werden.

Ergebnis

Der gesamte Bestellprozess an die Produktionszelle kann nun über die Bestell-App abgewickelt werden. Auch die Produktionsdaten können visuell dargestellt werden. Zusätzlich kann der Administrator die Bestellmöglichkeit aktivieren/deaktivieren. Steht die Produktionszelle gerade nicht zur Verfügung, kann alternativ eine Simulation für den Bestellprozess gestartet werden. Ein weiteres Highlight ist sicher der dreidimensionale Unihockeyball, welcher das Bestellerlebnis markant steigert. Filter in der Bestellübersicht und ein Responsive Design runden das Frontend ab.

Das Backend ist in drei logisch getrennte Container gegliedert. Ein Container übernimmt die Kommunikation zu MindSphere, während ein anderer Container die Daten vom (externen) Bestellsystem aufbereitet. Der dritte Container bindet alle Container und bietet gegenüber dem Frontend offene Schnittstellen an. Diese Aufteilung garantiert eine hohe Wartbarkeit.

Technologisch wurde für das Frontend auf Typescript und React gesetzt. Für das Backend kam C# beziehungsweise ASP.NET Core zum Einsatz. Da zum gegebenen Zeitpunkt keine SAP-Schnittstelle vorlag, wurde kurzerhand mit einem weiteren Container eine direkte Verbindung zwischen der Produktionszelle und Bestell-App realisiert. Durch die konsequente Containerisierung und Continuous Deployment liegt ein produktiv lauffähiger Release der Bestell-App bereit.

Inhaltsverzeichnis

1. Management Summary

1.1. Ausgangslage

Der Techpark der OST am Standort Rapperswil dient als Spielwiese für die angewandte Forschung und Entwicklung. Dort werden verschiedene Show-Cases für den Unterricht, als auch für Industriepartner entwickelt. Im Rahmen der Smart Factory, der Fabrik von morgen, wurde als neuster Show-Case eine Produktionszelle für das vollautomatisierte Herstellen von benutzerdefinierten Unihockeybällen erstellt.

Für diese wird nun im Verlauf dieser Studienarbeit eine Bestell-App entwickelt, welche über ein externes Bestellsystem die Ballproduktion ansteuern soll. Das Ziel der Arbeit ist eine Software von der Konzeptionierung bis zum getesteten Endprodukt vollständig zu entwickeln und zu dokumentieren. Es sollen verschiedene erlernte Methoden und Technologien aus anderen Modulen praktisch angewendet und neue Fachkenntnisse dazugewonnen werden. Ebenfalls sollen erste Erfahrungen in der Durchführung eines Softwareprojekts mit externen Stakeholdern gesammelt und der Aufbau einer akademischen Arbeit kennengelernt werden. Das erleichtert den Einstieg und die Absolvierung der Bachelorarbeit im kommenden Semester.

1.2. Vorgehen

Als erstes Produkt der Studienarbeit wurde ein Projektplan entwickelt, welcher die Projektorganisation im Team festgelegt. Ausserdem wurden die Zeitverhältnisse analysiert und Meilensteine definiert. Mit dem Projektplan wurde die Basis des Projektmanagements geschaffen.

In der Elaborationsphase wurde anschliessend die Umsetzung der Aufgabenstellung geplant. Ein wichtiger Bestandteil davon war das Zurechtlegen der Softwarearchitektur. Diese sollte eine reibungslose Implementierung, Veröffentlichung und Erweiterung ermöglichen. Zusätzlich wurden zu diesem Zeitpunkt die Anforderungen an das Produkt evaluiert.

Nachdem diese Voraussetzungen geschaffen wurden, konnte mit der Implementierung begonnen werden. Mit einem agilen Projektmanagement wurde zuerst ein Minimal Viable Product (MVP) entwickelt. Innerhalb von mehreren Iterationen wurde diese Basis anschliessend mit zusätzlichen Features erweitert. Danach wurden diese Ergebnisse im technischen Bericht dieser Arbeit dokumentiert.

1.3. Ergebnisse

Die Mehrheit der Anforderungen an das Produkt konnten in der Bestell-App realisiert werden. Lediglich die Anbindung an das externe Bestellsystem, konnte nicht realisiert werden. Für dessen Umsetzung werden Schnittstellen benötigt, welche bis zum Projektende nicht angeboten werden konnten. Aus diesem Grund wurde eine eigene interne Bestellungsverwaltung entwickelt, welche dessen Aufgaben übernimmt. Diese ermöglicht eine direkte Kommunikation mit der Ballproduktionsmaschine. Dank dieser kurzfristigen Ergänzung der Bestell-App konnten die restlichen Anforderungen an die Studienarbeit erfüllt werden.

Die finale Bestell-App ermöglicht die Ballkonfiguration und Bestellung eines Unihockeyballes. Im Konfigurator kann zwischen einer 2D als auch einer 3D Version des Balles gewechselt werden, welche die Farbauswahl des Nutzers demonstrieren. Bestellte Bälle werden in einer Übersicht aufgelistet, wo der aktuelle Zustand im Bestell- und Produktionsprozess ersichtlich ist. In dieser Übersicht werden diverse Filtermöglichkeiten angeboten. Es kann nach Farben und nach Bestellstatus gefiltert werden. Jede Bestellung verfügt über eine Detailansicht. Darin wird beispielsweise der QR-Code für produzierte und abholbereite Bälle angezeigt. Durch ihn können die fertigen Unihockeybälle bei der Produktionszelle abgeholt werden. Ausserdem befindet sich dort ein Link, welcher den Benutzer zu den Produktionsdaten dieses Unihockeyballes führt. Dort kann er die von Sensoren gemessenen Daten einsehen, welche beim Giessen und Verschweissen der Ballhälften erhoben wurden. Aufgrund eines technischen Defekts können zurzeit aber keine aktuellen Daten dargestellt werden. Der gesamte Funktionsumfang kann sowohl auf Desktop-, als auch mobilen Geräten wie Smartphones und Tablets genutzt werden.

1.4. Ausblick

Die entwickelte Bestell-App ist noch ausbaufähig. Als erste Erweiterung soll die Anbindung an das externe Bestellsystem der Smart Factory, welche aufgrund der fehlenden Schnittstellen bisher nicht implementiert werden konnte, umgesetzt werden. Derzeit werden die Produktionsdaten der Produktionszelle nicht in die Cloud gespeichert, die von der Bestell-App verwendet wird. Sobald der Upload wieder funktioniert, kann auch dieses Feature vervollständigt werden. Die Vorarbeiten für beide Anpassungen sind bereits durchgeführt und dokumentiert worden.

Um den Show-Case der Smart Factory interessanter zu gestalten, kann die Bestell-App mit weiteren Ideen ausgebaut zu werden. Die Möglichkeiten dafür sind endlos.

2. Anforderungsspezifikation

2.1. Use Cases

Für eine erste Übersicht der funktionalen Anforderungen werden Uses Cases erarbeitet. Diese sollen Aktoren und deren Rolle in den Prozessen der Bestell-App erläutern. Als Vorarbeit steht hier die Bachelorarbeit Smart Factory Konfigurator[1] von Robin Züllig aus dem Frühjahrsemester 2021 zur Verfügung, in welcher er in Kapitel 5.2.3 seine Use Cases im Brief Format erläutert. Im Anhang seiner Arbeit sind diese in der Fully Dressed Form genauer beschrieben. Im Rahmen dieses Kapitels werden diese Use Cases analysiert, falls nötig überarbeitet und ergänzt. Dadurch sollen die aktuellen Anforderungen der Stakeholder ersichtlich werden und später in das Produkt einfliessen.

2.1.1. Use Case Diagramm

In folgendem Use Case Diagram werden alle beschriebenen Use Cases und deren Beziehung mit den jeweiligen Akteuren visualisiert. Zusätzlich werden Abhängigkeiten zwischen den Use Cases dargestellt. Die Unterscheidung von MVP spezifischen Use Cases und solchen, die sich auf optionale Funktionalitäten beziehen, ist farblich ersichtlich.

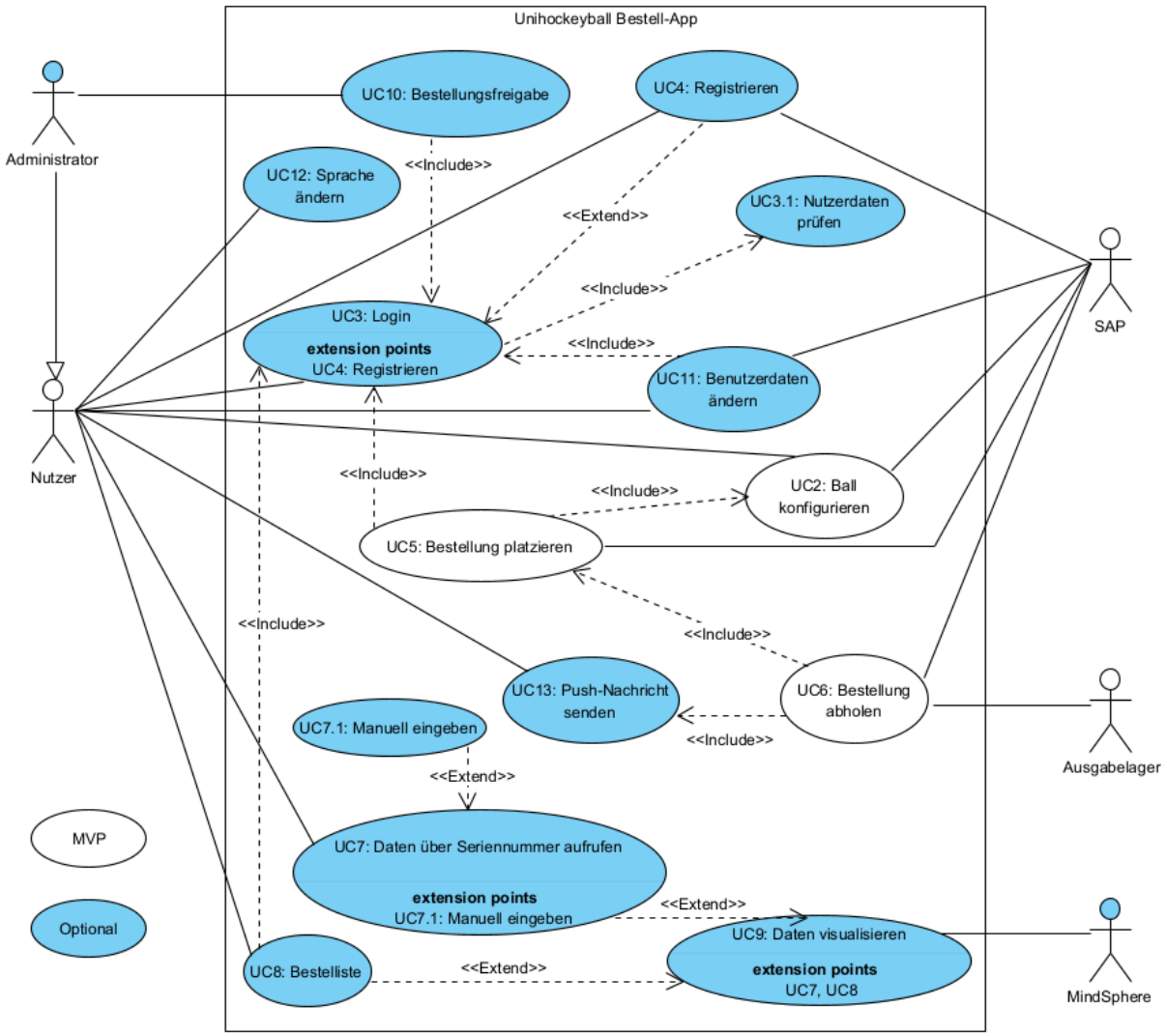

Abbildung 1: Use Case Diagramm

2.1.2. Vordefinierte Use Cases

Die vordefinierten Use Cases werden zuest im Brief Format von Robin Züllig zitiert. Danach werden falls nötig Ergänzungen und Anpassungen an den Fully Dressed Use Cases aufgelistet.

Use Case 1: App starten
Der Nutzer hat die Möglichkeit, vor Ort bei der smarten Produktionszelle einen QR-Code zu scannen und so den URL der App zu öffnen. Danach gelangt er über einen Browser auf die Startseite der Web App, wo er alle nötigen Informationen und Verlinkungen zu angesagten Funktionen erhält.[1]

Dieser Use Case hat weder Einfluss auf die Planung der Softwarearchitektur, noch auf die Umsetzung der Bestell-App und wird vernachlässigt. Unabhängig von der Implementation kann der beschriebene QR-Code generiert und bei der Produktionszelle platziert werden.

Use Case 2: Ball konfigurieren
Dem Nutzer wird auf der Startseite die Möglichkeit angeboten, seinen eigenen Ball zu konfigurieren. Beim Anwählen des Konfigurators wird er auf eine neue Seite weitergeleitet, wo er aus einer Liste möglicher Ballhälften sein eigenes Wunschprodukt zusammenstellen kann.[1]

Grundsätzlich ist der Use Case passend und signifikant für die Umsetzung. Folgende Punkte werden angepasst:

Abschnitt	Änderung	Begründung
Vorbedingungen	Hinfällig	Der Konfigurator ist auch direkt über die URL erreichbar.
Haupterfolgsszenario Punkt 7	Ballhälften welche nicht an Lager sind, werden angezeigt und als nicht verfügbar gekennzeichnet.	Bessere Usability, da der Nutzer die Verfügbarkeit besser nachvollziehen kann.

Tabelle 1: Anpassung Use Case 2

Use Case 3: Login
Der Nutzer hat bereits zu einem früheren Zeitpunkt einen Nutzeraccount erstellt und gelangt über den Button «Login» zur Login-Seite. Dort kann er sich anhand seiner E-Mail identifizieren.[1]

Grundsätzlich ist der Use Case passend und signifikant für die Umsetzung. Folgende Punkte werden angepasst:

Abschnitt	Änderung	Begründung
Vorbedingungen	Hinfällig	Die Login Page ist auch direkt über die URL erreichbar.
Spezielle Anforderungen	Die Sprachauswahl in der WebApp wird in einen eigenen Use Case ausgelagert und die letzte Anforderung wird vernachlässigt.	Die Sprachauswahl weist keine Korrelation mit dem Login-Prozess auf. Letzteres ist keine funktionale Anforderung und wird in den Qualitätsanforderungen erläutert.

Tabelle 2: Anpassung Use Case 3

Use Case 4: Registrieren
Falls der Nutzer noch keinen Account hat, wird er über den Button «Registrieren» auf die Registrationsseite weitergeleitet, wo er Name, Vorname und E-Mail-Adresse angeben kann um sich zu registrieren. Das System leitet die Daten an das SAP weiter und gibt die Erstellung eines neuen Debitors in Auftrag. Nach der Erstellung des Debitor Kontos übergibt das SAP die neu erstellte Kundennummer an das System der SF App.[1]

Grundsätzlich ist der Use Case passend und signifikant für die Umsetzung. Folgende Punkte werden angepasst:

Abschnitt	Änderung	Begründung
Erfolgsgarantie	Die Anschrift und das Land werden bei der Registrierung nicht benötigt. Lediglich Name, Vorname Email und Passwort müssen angegeben werden.	Für den Bestellprozess werden keine Adressinformationen benötigt, da der Ball bei der Produktionszelle abgeholt wird.

Tabelle 3: Anpassung Use Case 4

Use Case 5: Bestellung platzieren
Nachdem der Nutzer eingeloggt ist und einen Ball konfiguriert hat, kann er diesen, indem er auf den Button «Bestellen» drückt, direkt im Konfigurator bestellen. Das System sendet zuerst eine Anfrage an das SAP, ob die gewählten Ballhälften vorhanden sind. Danach wird die Materialnummer und die Kundennummer an das SAP übergeben und eine Bestellung in Auftrag gegeben.[1]

Der Use Case ist passend und signifikant für die Umsetzung.

Use Case 6: Bestellung abholen
Nachdem der Nutzer einen Ball bestellt hat, wartet das System auf die Bestätigung vom SAP, dass der Ball im Ausgabelager liegt. Danach wird von der App eine Mitteilung an den Nutzer gesendet, welche ihn auffordert, den Ball mit dem beigelegten QR-Code beim Ausgabelager abzuholen. Der Nutzer holt darauf den bestellten Ball mit dem dem Abhol-QR-Code ab, welcher auf dem App visualisiert wird.[1]

Grundsätzlich ist der Use Case passend und signifikant für die Umsetzung. Folgende Punkte werden angepasst:

Abschnitt	Änderung	Begründung
Haupterfolgsszenario Punkt 2	Der Nutzer muss nicht explizit auf einen Button "Abholen" klicken.	Der QR-Code wird bereits bei der Bestellübersicht dargestellt, um die Komplexität der App zu minimieren.
Alternative Flüsse	Der gesammte Bestellprozess ist an das SAP gekoppelt. Fällt die Kommunikation aus, ist der Prozess vorübergehend unterbrochen und der Benutzer wird informiert.	Um die Integrität der Bestellung gewährleisten zu können, dürfen keine Aktionen ohne das SAP ausgeführt werden.

Tabelle 4: Anpassung Use Case 6

Use Case 7: Daten über Seriennummer aufrufen
Auf der Startseite besteht die Möglichkeit, eine beliebige Seriennummer auf einem Ball oder einer Ballhälfte über die Kamera des Devices zu scannen oder manuell einzugeben. Danach wird die Seriennummer im System aufgenommen und der Nutzer gelangt zu Use Case 9.[1]

Der Use Case ist passend und signifikant für die Umsetzung.

Use Case 8: Bestellliste
Jeder Nutzer hat die Möglichkeit in seiner Bestellliste alle seine vergangenen und aktuell laufenden Bestellungen anzusehen. Dazu kann er von den einzelnen Bestellungen die Daten anzeigen lassen oder die Abholnachricht aufrufen.[1]

Der Use Case ist passend und signifikant für die Umsetzung. Folgende Punkte werden angepasst:

Abschnitt	Änderung	Begründung
Haupterfolgsszenario Punkt 3	Ist die Produktion des Balles noch nicht abgeschlossen, wird eine voraussichtliche Wartezeit angezeigt.	Erweiterung des Use Cases

Tabelle 5: Anpassung Use Case 8

Use Case 9: Daten visualisieren
Falls über Use Case 7 oder Use Case 8 auf die Daten einer spezifischen Seriennummer zugegriffen werden möchte, sendet das System eine Anfrage an die MindSphere, um diese Daten zu erhalten. Anschliessend präsentiert das System die Daten auf eine möglichst verständliche Art.[1]

Grundsätzlich ist der Use Case passend und signifikant für die Umsetzung. Folgende Punkte werden angepasst:

Abschnitt	Änderung	Begründung
Alternative Flüsse Punkt 2b	Falls MindSphere die Daten nicht liefern kann, wird eine Fehlermeldung dargestellt.	Die weiteren Beschreibungen sind Qualitätsanforderungen und werden als solche in den Nichtfunktionalen Anforderungen beschrieben.

Tabelle 6: Anpassung Use Case 9

Use Case 10: Bestellungsfreigabe
Durch die Eingabe einer definierten E-Mail oder einem String im Login Prozess gelangt der Zugang zur Administratorfunktion. Der Administrator hat die zusätzliche Möglichkeit, die Bestellfunktion zu aktivieren oder zu deaktivieren. Somit können die Nutzer nur eine Bestellung aufgeben, wenn der Administrator die Bestellfunktion auch freigegeben hat. Dies sorgt dafür, dass nicht zu jeder Zeit und in geordnetem Mass Bestellungen bei der Produktionszelle eingehen.[1]

Grundsätzlich ist der Use Case passend und signifikant für die Umsetzung. Folgende Punkte werden angepasst:

Abschnitt	Änderung	Begründung
Name	Der Use Case wurde von Administrationsfunktion zu Bestellungsfreigabe umbenannt.	Besseres Verständnis und Einheitlichkeit
Vorbedingungen	Hinfällig	Die Login Page ist auch direkt über die URL erreichbar.

Tabelle 7: Anpassung Use Case 10

2.1.3. Zusätzliche Use Cases

Um den funktionellen Umfang der Bestell-App genauer zu spezifizieren, wurden die folgenden Use Cases im Fully Dressed Format selbst erarbeitet.

Use Case 11: Benutzerdaten ändern
Ein eingeloggter Nutzer kann auf eine Profilübersicht navigieren. Dort hat er die Möglichkeit, in einen Bearbeitungsmodus zu gelangen, um seine Angaben zu ändern. Seine Anpassungen kann er anschliessend speichern.

Abschnitt	**Kommentar**
Name	Benutzerdaten ändern
Beschreibung	Ein eingeloggter Benutzer kann auf der Profilübersicht seine Daten einsehen und hat die Möglichkeit diese zu ändern
Primärer Akteur	Nutzer (Kunde)
Stakeholder und Intressegruppen	Nutzer: Will seine Angaben ändern SAP: Aktualisiert die Benutzerdaten
Vorbedingungen	Nutzer muss authentifiziert sein
Erfolgsgarantie	Nutzerdaten sind sowohl in der Bestell-App, als auch im SAP aktualisiert und einheitlich
Haupterfolgsszenario	1. Der Nutzer navigiert von der Startseite auf die Übersicht seines Profils 2. Der Nutzer wechselt in den Bearbeitungsmodus 3. Der Nutzer bearbeitet seine Daten 4. Der Nutzer speichert seine Änderungen 5. Die Bestell-App überprüft, ob die Angaben des Nutzers den Vorgaben entsprechen 6. Die Änderungen werden in der Datenbank der Bestell-App gespeichert 7. Die Bestell-App sendet dem SAP die aktualisierten Nutzerdaten mit dem Auftrag, diese zu speichern 8. Das SAP informiert die Bestell-App über die erfolgreiche Änderung der Nutzerdaten 9. Die Bestell-App informiert den Nutzer über die erfolgreiche Änderung und zeigt die aktualisierten Daten an
Alternative Flüsse	5a: Erfüllen die Angaben des Benutzers die Vorgaben nicht, wird eine Fehlermeldung dargestellt 9a: Ist Punkt 6 oder Punkt 8 fehlgeschlagen, wird die Aktualisierung abgebrochen und der Benutzer entsprechend informiert

Tabelle 8: Use Case 11

Use Case 12: Sprache ändern
Der Nutzer hat auf jeder Seite der Bestell-App die Möglichkeit zu der Sprachauswahl zu navigieren. Dort kann er zwischen verschiedenen Sprachen auswählen. Nach seiner Auswahl, werden alle Texte der Bestell-App in der gewünschten Sprache dargestellt.

Abschnitt	**Kommentar**
Name	Sprache ändern
Beschreibung	Ein Nutzer ändert die Sprache der Bestell-App
Primärer Akteur	Nutzer (Kunde)
Stakeholder und Intressegruppen	Nutzer: Will die Sprache der Bestell-App ändern
Vorbedingungen	-
Erfolgsgarantie	Die Bestell-App wird in der gewünschten Sprache des Nutzers angezeigt
Haupterfolgsszenario	1. Der Nutzer navigiert von einer beliebigen Seite auf die Sprachauswahl 2. Der Nutzer wählt eine der verfügbaren Sprachen aus 3. Die Bestell-App wird in der gewünschten Sprache angezeigt 4. Die Auswahl des Nutzers wird in seiner Session gespeichert 5. Bei zukünftigen Besuchen in einem bestimmten Zeitraum wird die Sprachauswahl berücksichtigt, ansonsten wird sie auf Englisch eingestellt
Alternative Flüsse	-

Tabelle 9: Use Case 12

Use Case 13: Push-Nachricht senden

Abschnitt	Kommentar
Name	Push-Nachricht senden
Beschreibung	Sobald ein Ball produziert und abholbereit ist, wird der Nutzer mittels einer Push-Nachricht darüber informiert
Primärer Akteur	Bestell-App
Stakeholder und Intressegruppen	Nutzer: Will über die Abholbereitschaft informiert werden
Vorbedingungen	Bestellung wurde in Auftrag gegeben Der Ball wurde produziert Der Nutzer ist authentifiziert
Erfolgsgarantie	Der Nutzer erhält nach erfolgreicher Produktion des Balles eine Push-Nachricht
Haupterfolgsszenario	1. Das SAP informiert die Bestell-App über eine abgeschlossene Fertigung 2. Die Bestell-App ermittelt den betreffenden Nutzer 3. Die Bestell-App sendet dem Nutzers eine Push-Nachricht
Alternative Flüsse	-

Tabelle 10: Use Case 13

2.1.4. MVP

Die Anforderungen an das MVP wurden so definiert, dass ein Nutzer erfolgreich einen Unihockeyball bestellen und bei der Produktionszelle abholen kann. Für die Bestell-App bedeutet dies, dass sich ein Nutzer registrieren und einloggen kann. Er muss in der Lage sein, die Farben der zwei Ballhälften zu konfigurieren und danach eine Bestellung in Auftrag zu geben. Für diese Umsetzung ist die Anbindung an das SAP eine essentielle Voraussetzung. Das SAP informiert die Bestell-App über Statusänderungen, damit diese dem Nutzer die Abholung des Balles mit einem QR-Code ermöglicht.

2.1.5. Priorisierung der optionalen Use Cases

Auch die optionalen Use Cases sind ein wichtiger Bestandteil einer benutzerfreundlichen und interessanten Bestell-App. Daher werden im Rahmen der Studienarbeit möglichst viele davon implementiert. Für die Reihenfolge deren Umsetzung wurden sie anhand ihres Mehrwertes priorisiert:

1. UC3: Login
2. UC4: Registrieren
3. UC8: Bestellliste
4. UC9: Daten visualisieren
5. UC7 / UC7.1: Daten über Seriennummer aufrufen
6. UC10: Bestellungsfreigabe
7. UC13: Push-Nachricht senden
8. UC11: Benutzerdaten ändern
9. UC12: Sprache ändern

2.2. Feature spezifische Requirements

Im Kapitel 5.2.6 Software Requirements Specification der Bachelorarbeit[1] von Robin Züllig werden Features des Produktes, welche anhand der Use Cases erarbeitet wurden, aufgezählt. Jedoch werden sie erst im Anhang der Arbeit genauer beschrieben. Dort sind zu jedem Feature sowohl funktionale, als auch nichtfunktionale Anforderungen beschrieben.

Die funktionalen Anforderungen werden nun im Rahmen dieser Arbeit analysiert und falls nötig so angepasst, dass sie den Bedürfnissen eines Softwareentwicklers entsprechen. Sie dienen in der Implementationsphase des Projektes als Voraussetzung und Kontrollmass.

Die Labels von Buttons werden nicht wie von Robin Züllig in den Feature Anforderungen beschrieben, sondern erst in der Designphase des Bestell-Apps definiert. Der Fokus liegt zu diesem Zeitpunkt auf den Funktionalitäten der Features.

2.2.1. Startseite

ID	Feature Name	Anpassung	Begründung
FR-HP1	Button: Datenvisualisierung	-	-
FR-HP2	Button: Konfigurator	-	-
FR-HP3	Button: Login	-	-
FR-HP4	Button: Registrierung	Der Registrierungsbutton wird immer dargestellt.	Der Nutzer soll immer die Möglichkeit haben, ein neues Konto anzulegen.
FR-HP5	Button: Bestellliste	Nur sichtbar, falls der Nutzer eingeloggt ist.	Usability
FR-HP6	Button: App Übersicht	-	-
FR-HP7	Button: Profil	Eingeloggte Nutzer können eine Kontoübersicht aufrufen.	Ergänzung

Tabelle 11: Features Startseite

2.2.2. Konfigurator

ID	Feature Name	Anpassung	Begründung
FR-CR1	Konfigurator Modell	-	-
FR-CR2	Farbe wählen	-	-
FR-CR3	Button: Bestellen	Der Button ist erst aktiv, wenn zwei Ballhälften ausgewählt sind und der Nutzer eingeloggt ist.	Usability
FR-CR4	Button: Zurück	-	-

Tabelle 12: Features Konfigurator

2.2.3. Login

ID	Feature Name	Anpassung	Begründung
FR-LI1	Login	Zusätzlich zur Email wird auch ein Passwort benötigt.	Wird für die Authentifizierung vorausgesetzt.
FR-LI2	Button: Abbrechen	-	-

Tabelle 13: Features Login

2.2.4. Registrierung

ID	Feature Name	Anpassung	Begründung
FR-RI1	Button: Registrierung	-	-
FR-RI2	Nutzerangaben	Für die Registrierung muss der Nutzer nur Vorname, Name, Email und Passwort angeben.	Für den Bestellprozess werden keine Adressinformationen benötigt. Die Sprachauswahl ist ein eigenständiges Feature.
FR-RI3	SAP Daten	Nicht berücksichtigt	Ist kein eigenes Feature sondern Bestandteil von FR-RI4
FR-RI4	Debitor erstellen	-	-
FR-RI5	Button: Zurück	-	-

Tabelle 14: Features Registrierung

2.2.5. Bestellung

ID	Feature Name	Anpassung	Begründung
FR-PO1	Bestätigung	-	-
FR-PO2	SAP Bestellung eröffnen	-	-

Tabelle 15: Features Bestellung

2.2.6. Bestellübersicht

ID	Feature Name	Anpassung	Begründung
FR-PU1	Produktionsstatus	Sobald die Produktion abgeschlossen ist, wird ein QR-Code für die Abholung dargestellt.	Ergänzung
FR-PU2	Button: Datenvisualisierung	Verschobenes Feature aus Bestellliste	Usability
FR-PU3	Abholaufforderung	-	-

Tabelle 16: Features Bestellübersicht

2.2.7. Datenvisualisierung

ID	Feature Name	Anpassung	Begründung
FR-DV1	Button: Datenvisualisierung	-	-
FR-DV2	Seriennummer scannen	-	-
FR-DV3	Manuelle Eingabe	-	-
FR-DV4	Keine Kamera	Nicht berücksichtigt	Technisch nicht umsetzbar
FR-DV5	Darstellung der Daten	-	-

Tabelle 17: Features Datenvisualisierung

2.2.8. Bestellliste

ID	Feature Name	Anpassung	Begründung
FR-OL1	Button: Bestellliste	-	-
FR-OL2	Button: Abholung	Nicht berücksichtigt	Der QR-Code wird direkt in der Bestellübersicht angezeigt.
FR-OL3	Button: Datenvisualisierung	Feature wird in die Bestellübersicht verschoben	Usability
FR-OL4	Produktionsstatus	-	-

Tabelle 18: Features Bestellliste

2.2.9. Bestellungsfreigabe

ID	Feature Name	Anpassung	Begründung
FR-AD1	Identifikation	-	-
FR-AD2	Button: Einstellungen	Der Button führt zu einer Einstellungsseite.	Ergänzung
FR-AD3	Einstellungen	-	-

Tabelle 19: Features Bestellungsfreigabe

2.2.10. Benutzerdaten

ID	Feature Name	Beschreibung
FR-BA1	Anzeige	Die Nutzerangaben werden dargestellt.

FR-BA2	Button: Bearbeiten	Über den Button gelangt der Nutzer in den Bearbeitungsmodus.
FR-BA3	Button: Speichern	Im Bearbeitungsmodus kann der Nutzer seine Änderungen speichern.
FR-BA4	Button: Abbrechen	Der Bearbeitungsmodus wird verlassen und die Änderungen verworfen.

Tabelle 20: Features Benutzerdaten

2.2.11. Sprachauswahl

ID	Feature Name	Beschreibung
FR-SA1	Button: Sprachauswahl	Auf jeder Seite hat der Nutzer die Möglichkeit die Sprache der Bestell-App zu ändern.

Tabelle 21: Features Sprachauswahl

2.3. Nichtfunktionale Anforderungen (NFA)

Die nicht funktionalen Anforderungen der Unihockeyball Bestell-App werden anhand des ISO-25010[2] Standards für Softwarequalität aufgeführt. Als Ausgangslage dient erneut die Bachelorarbeit[1] von Robin Züllig, in welcher die NFA im Anhang ohne vermerkten Standard beschrieben sind.

2.3.1. Functional Suitability

Kriterium	Beschreibung
Functional Completeness	Jeder Benutzer kann, die in diesem Dokument beschriebenen und schlussendlich implementierten, Use Cases vollständig ausführen.
Functional Correctness	Dem Kunden werden nur eigene Bestellungen angezeigt.
	Der Kunde kann nur Farben an Lager auswählen.
	Sämtliche Abhängigkeiten und Bedingungen zwischen einzelnen Use Cases (Bsp. Bestellung erfordert Authentifizierung) sind aufgelöst und eingehalten.

Tabelle 22: NFA Suitability

2.3.2. Performance efficiency

Kriterium	Beschreibung
Time Behaviour	Eine erfasste Bestellung wird innerhalb von 5 Sekunden verarbeitet und gespeichert (an das SAP geliefert).
	Ein Benutzer wird innerhalb von 5 Sekunden eingeloggt.
Capacity	Die Applikation kann mit 168 simultan aktiven Benutzern umgehen, ohne die Zeitanforderungen zu überschreiten. *Zahlen gemäss BA von Robin Züllig*

	Die Applikation kann jährlich mindestens 10'000 Bestellung verarbeiten und dessen kumulierenden Daten speichern.
	Die Applikation kann jährlich 2178 neue Benutzerkonten aufnehmen. *Zahlen gemäss BA von Robin Züllig*

Tabelle 23: NFA Performance efficiency

2.3.3. Compatibility

Kriterium	Beschreibung
Interoperability	Füe die Kommunikation zwischen Softwarekomponenten werden standardisierte Schnittstellen verwendet.

Tabelle 24: NFA Compatibility

2.3.4. Usability

Kriterium	Beschreibung
Appropriateness recognizability	Auf der Startseite bekommt der Benutzer einen Überblick über den Funktionsumfang der Bestell-App.
Operability	Der gesamte Funktionsumfang kann mittels Touchscreen genutzt werden. Dafür sind Interaktionselemente (Bsp. Button) mindestens 32 Pixel breit.
	Die Web-Applikation wird für folgende Bildschirmgrössen übersichtlich dargestellt: • Smartphone: 390x844 Pixel (IPhone 12) • Tablet: 1194x834 Pixel (IPad 11) • Desktop: 1920x1080 Pixel (Meistverwendete Monitore) Die Tests werden mit diesen Werten durchgeführt. Die Bestell-App soll aber auch bei Abweichungen übersichtlich sein.
	Beim Login wird dem Benutzer ein Token ausgestellt, mit welchem er sich später wieder authentifizieren kann.
User error protection	Beim Bestellprozess muss der Nutzer die Bestellung bestätigen.
	Der Nutzer wird bei ausserordentlichem Verhalten der Bestell-App (Bsp. Ausfall von Subsystemen) durch eine passende Fehlermeldung informiert.
Accessibility	Statusinformationen werden nicht nur mit Farbe dargestellt, sondern auch mit Symbolen/Text.
	Ein Farbkontrast von mindestens 5.7:1 (Level AA) wird auf dem gesamten UI gewährleistet.
	Das UI erfüllt den Google Chrome Accessibility Audit[3].
	Die Webseite ist zoombar.

Tabelle 25: NFA Usability

2.3.5. Reliability

Kriterium	Beschreibung
Availability	Die Bestell-App ist während Show-Cases erreichbar, sofern es die OST-Infrastruktur zulässt.
Fault tolerance	Die Benutzereingaben werden bereits im Frontend validiert, wie zum Beispiel, dass bei der Registrierung keine Mailadresse ohne @-Zeichen akzeptiert wird.
	Bei ungültiger Benutzereingabe wird dem Nutzer eine Fehlermeldung angezeigt.
	Falsche Eingaben vom Benutzer auf der Webseite haben keine Auswirkungen auf die Verfügbarkeit der Dienstleistung.
	Fallen externe Schnittstellen aus, sind betroffene Features deaktiviert und der Nutzer wird darüber informiert.

Tabelle 26: NFA Reliability

2.3.6. Security

Kriterium	Beschreibung
Confidentiality	Das Passwort wird als Hash-Code gespeichert.
	Logs enthalten keine vertraulichen Informationen (Bsp. Benutzerdaten).
	Die Kommunikation zwischen einzelnen Softwarekomponenten und den externen Schnittstellen ist mit TLS verschlüsselt.
Accountability	Jede Bestellung ist einem Account und somit einem Debitor zuweisbar.
Authenticity	Der Zugriff auf Benutzerdaten verlangt die Authentifizierung mittels Email und Passwort.

Tabelle 27: NFA Security

2.3.7. Maintainability

Kriterium	Beschreibung
Modularity	Softwarekomponenten werden in einzelnen Containern implementiert und deployed.
	Die Softwarearchitektur wird in mehreren Tiers umgesetzt, um einzelne Komponenten austauschbar zu machen.
Analysability	Der Programmcode muss verständlich sein.
	Komplexe Design-Entscheidungen sind zu dokumentieren.
	Definierte Style-Guidelines werden eingehalten.
Testability	Das Software Design ermöglich ein einfaches Mocking von Bestandteilen.
	Unit Tests werden mit der CI/CD Pipeline von GitLab automatisiert und bei jeder Änderung im Respository ausgeführt.

Tabelle 28: NFA Maintainability

2.3.8. Portability

Kriterium	Beschreibung
Adaptability	Die Web-Applikation wird auf den gängigsten Browser korrekt dargestellt.
	Der Funktionsumfang ist ab den aufgeführten Browser-Versionen uneingeschränkt verfügbar: • Google Chrome: Version 58+ • Firefox: Version 54+ • Safari: Version 10.1+
Installability	Das Produkt kann in Containern (Bsp. Docker) bereitgestellt werden.
Replaceability	Die SAP Anbindung wird bestmöglich abstrahiert damit diese falls nötig möglichst effizient umgehen werden kann.

Tabelle 29: NFA Portability

2.4. Domain Model

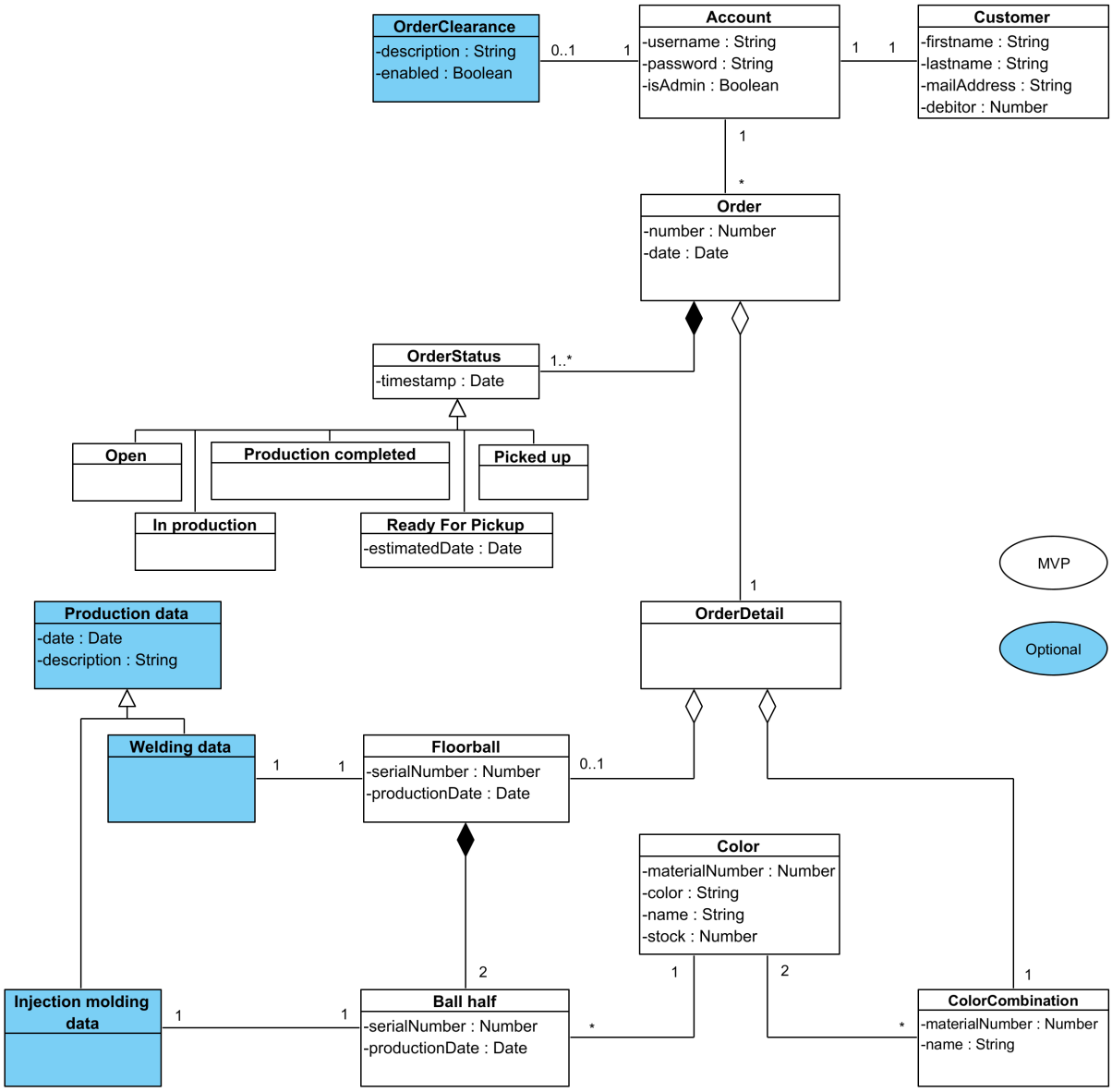

Abbildung 2: Domain Model

Bestandteil	Beschreibung
Account	Ein *Account* beinhaltet sämtliche Informationen, welche für ein erfolgreiches Login beziehungsweise eine erfolgreiche Authentifizierung benötigt wird. Zusätzlich kann ein Administratoraccount als solcher erkannt werden.
Customer	Ein Nutzer, der eine neue Bestellung eröffnen und bestehende Bestellungen verwalten möchte, benötigt ein Profil in der Rolle *Customer*. SAP verlangt bei jeder Bestellung eine Debitorennummer des Kunden.
OrderClearance	Die Bestellmöglichkeit des *Customer* kann vorübergehend deaktiviert werden.

Order	Eine sehr zentrale Klasse in dieser Domäne. Sie bindet vom Fertigungsauftrag bis zum endgültigen Lieferobjekt alle Entitäten, welche im entsprechenden Bestellprozess geschaffen werden.
OrderStatus	Eine Bestellung durchläuft mehrere unabhängige Schritte. Jeder Schritt wird durch einen Status abgebildet und protokolliert der zeitlichen Verlauf.
OrderDetail	*OrderDetail* hält die Informationen, welche an die Fertigungszelle für die Produktion übermittelt werden (*Material*) und von dieser zurückgemeldet werden (*Item*, *Product*).
Color	Bildet die Farbe eines konkreten Balles oder einer Farbkombination ab. Zusätzlich wird der Lagerbestand der Farben aufgeführt.
Floorball	Das Lieferobjekt, der Unihockeyball, besteht aus zwei Ballhälften und besitzt eine eindeutige Seriennummer. Vertretet somit die konkrete Ball-Instanz.
Ball Half	Jede Ballhälfte entspricht einer physisch existierenden, einzigartigen Ballhälfte und besitzt eine eindeutige Seriennummer.
ColorCombination	Stellt das abstrakte Produkt im Bestellprozess dar, welches aus zwei Farben besteht.
Production data	Die einzelnen Fertigungschritte des Unihockeyballes liefern Produktionsdaten. Diese werden in Siemens MindSphere abgespeichert.
Welding data	Für einen Unihockeyball müssen zwei Ballhälften zusammengeschweisst werden.
Injection molding data	Jede Ballhälfte wird von einer Spritzgussmaschine gegossen.

Tabelle 30: Domain Model - Bestandteile

3. Design

3.1. Systemarchitektur

3.1.1. Systemübersicht

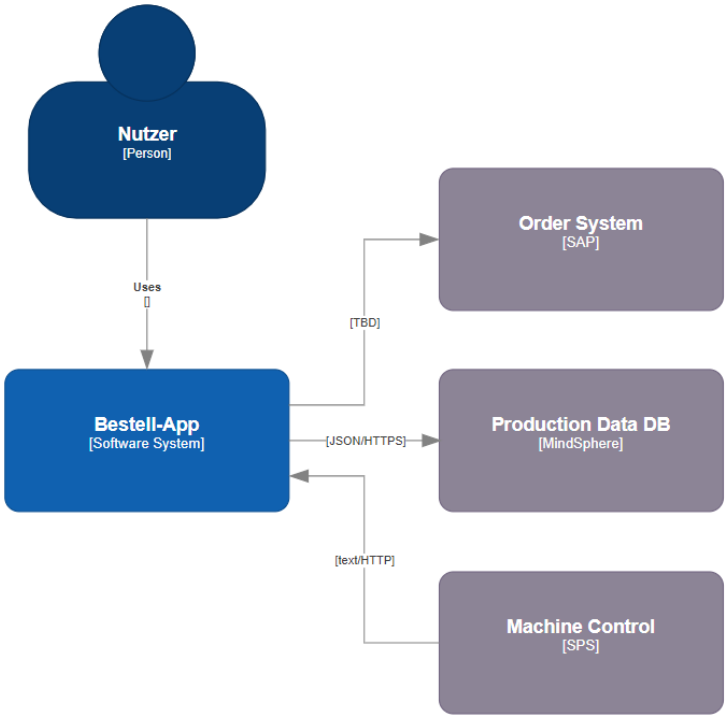

Abbildung 3: Systemübersicht

Die Bestell-App kommuniziert insgesamt mit vier Akteuren. Der Nutzer interagiert über ein Benutzerinterface mit der App und kann somit seine gewünschten Aktionen tätigen. Diese werden von der Bestell-App verarbeitet und falls nötig externe Systeme involviert. Für die Konfiguration und Bestellung eines Balles wird ein externes Bestellsystem (voraussichtlich SAP) verwendet, welches über eine bisher unbekannte Schnittstelle erreichbar sein sollte. Über eine externe Cloud (MindSphere) werden Produktionsdaten eines Balles abgerufen. Diese Kommunikation läuft über das HTTPS Protokoll und die Daten werden in JSON übertragen. Im Verlauf der Studienarbeit wurde zusätzlich eine direkte Anbindung an die Maschinensteuerung der Smart Factory implementiert, da keine Schnittstellen seitens SAP rechtzeitig zur Verfügung gestellt werden konnten.

3.1.2. Containerübersicht

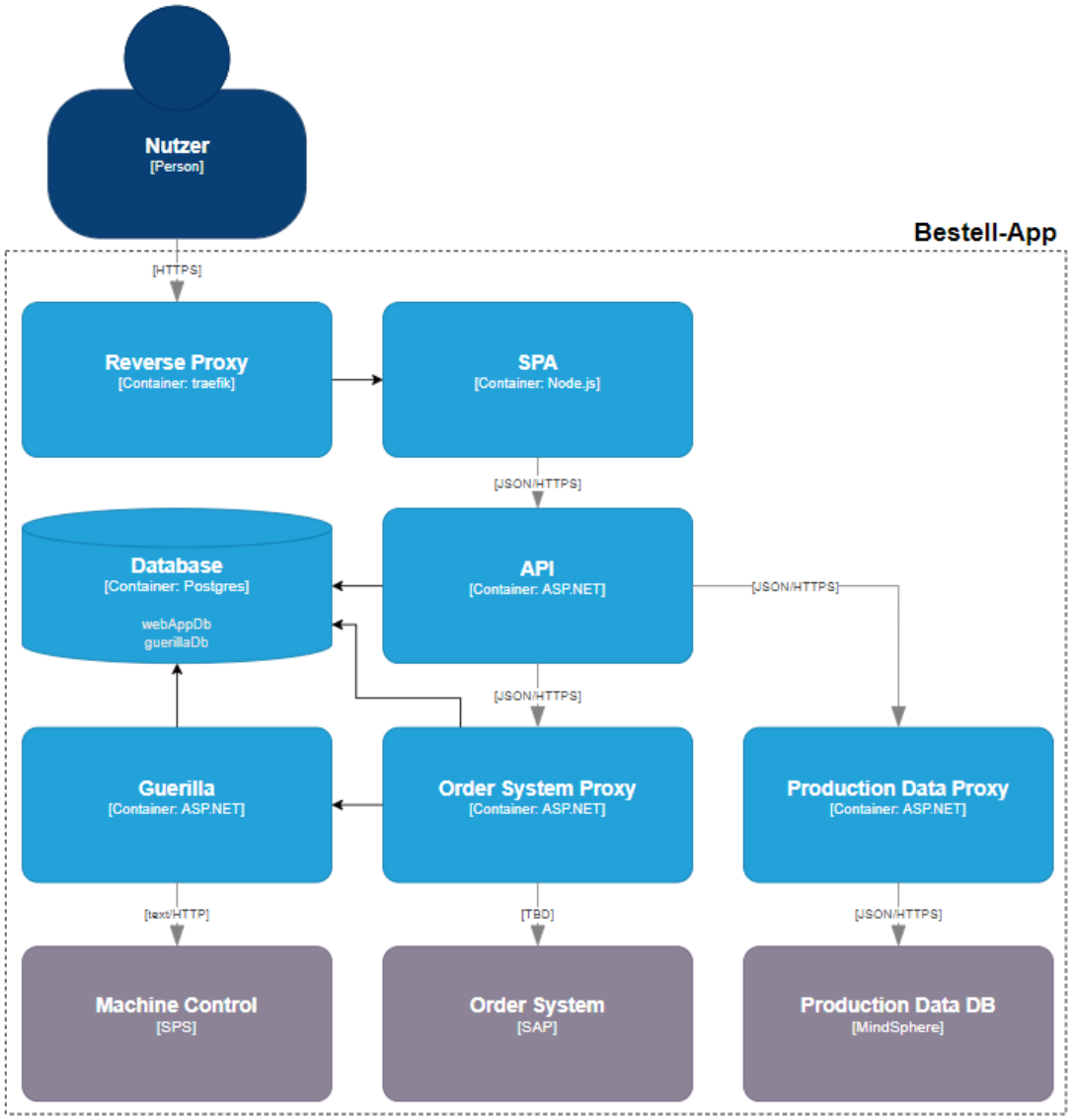

Abbildung 4: Containerübersicht

In der Containerübersicht wird der interne Aufbau der Bestell-App dargestellt. Der Nutzer interagiert indirekt über einen Reverse Proxy mit einem Node-Webserver, welcher ihm eine SPA ausliefert. Über diese Benutzeroberfläche sind dem Nutzer die beschriebenen Use Cases möglich. Seine Aktionen werden als HTTPS-Anfragen an einen API-Container gesendet, wo diese bearbeitet werden. Für Daten, welche persistiert werden müssen, wird ein PostgreSQL Container mit einer Datenkank verwendet. Anfragen, die auf externe Komponenten angewiesen sind, werden über REST Schnittstellen an Proxy-Container weitergeleitet, welche die externen Systeme abstrahieren.

3.1.3. Frontend

Das Frontend wird als SPA mit React entwickelt. Für zusätzliche Typensicherheit wird TypeScript als Programmiersprache verwendet. Optional kann die Bestell-App zu einer Progressive Web-App erweitert werden, um eine Installierbarkeit und gewisse Offline-Funktionalitäten gewährleisten zu können. Man hat sich jedoch gegen eine Umsetzung mit React Native entschieden, da das Produkt primär als Web-App verwendet wird. Somit besteht kein Bedarf, die Bestell-App zusätzlich als Mobile-App anzubieten.

Bereich	**Technologie**
UI-Library	React
Sprache	TypeScript
Webserver	Node
Testing	Testing Library und Jest
Styling	SASS
State Management	Redux
HTTP Client	Axios

Tabelle 31: Frontend Technologien

3.1.4. API

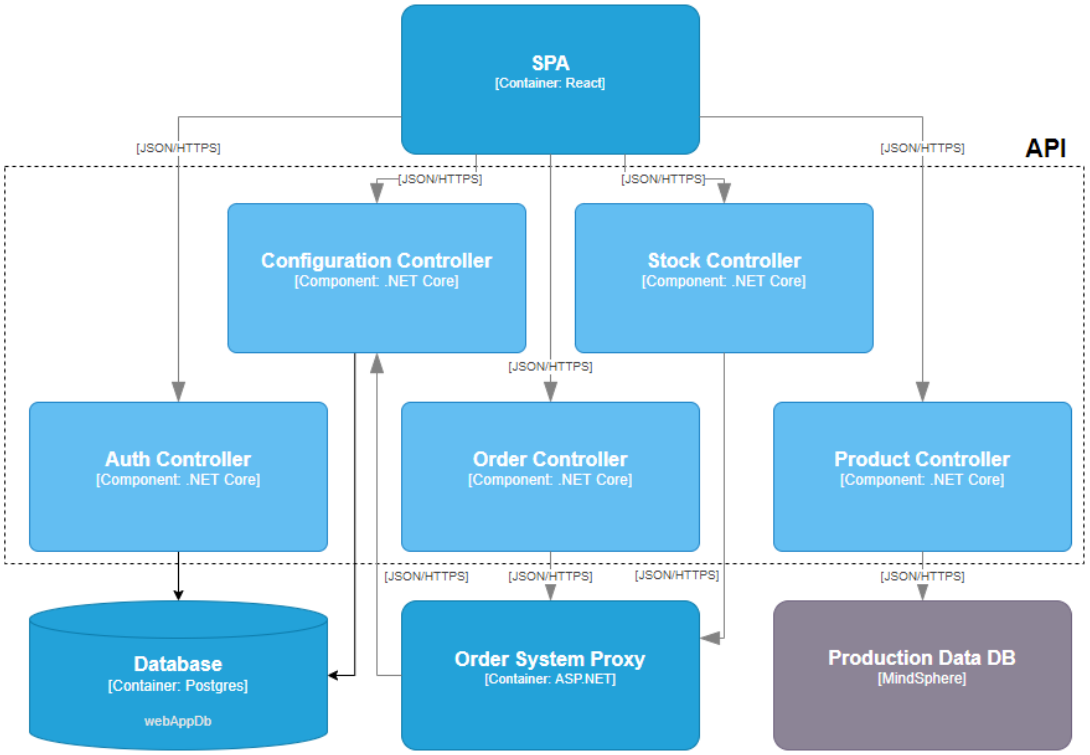

Abbildung 5: API

Für die Implementation des API-Containers wurde ASP.NET als Technologie ausgewählt, da sie sich sowohl für die Implementation der Businesslogik und der Schnittstellen eignet. Zusätzlich bietet eine .NET Umgebung die Möglichkeit, Entity Framwork Core als OR-Mapper für die Datenbank zu verwenden. Der API-Container sorgt sich um die Authentifizierung und ist verantwortlich, die Anfragen vom Frontend zu bearbeiten und an die korrekte Stelle weiterzuleiten.

Bereich	Technologie
Framework	ASP.NET
Sprache	C#
OR-Mapper	Entity Framework Core
Testing	xUnit.net und Fluent Assertions

Tabelle 32: API Technologien

3.1.5. Order System Proxy

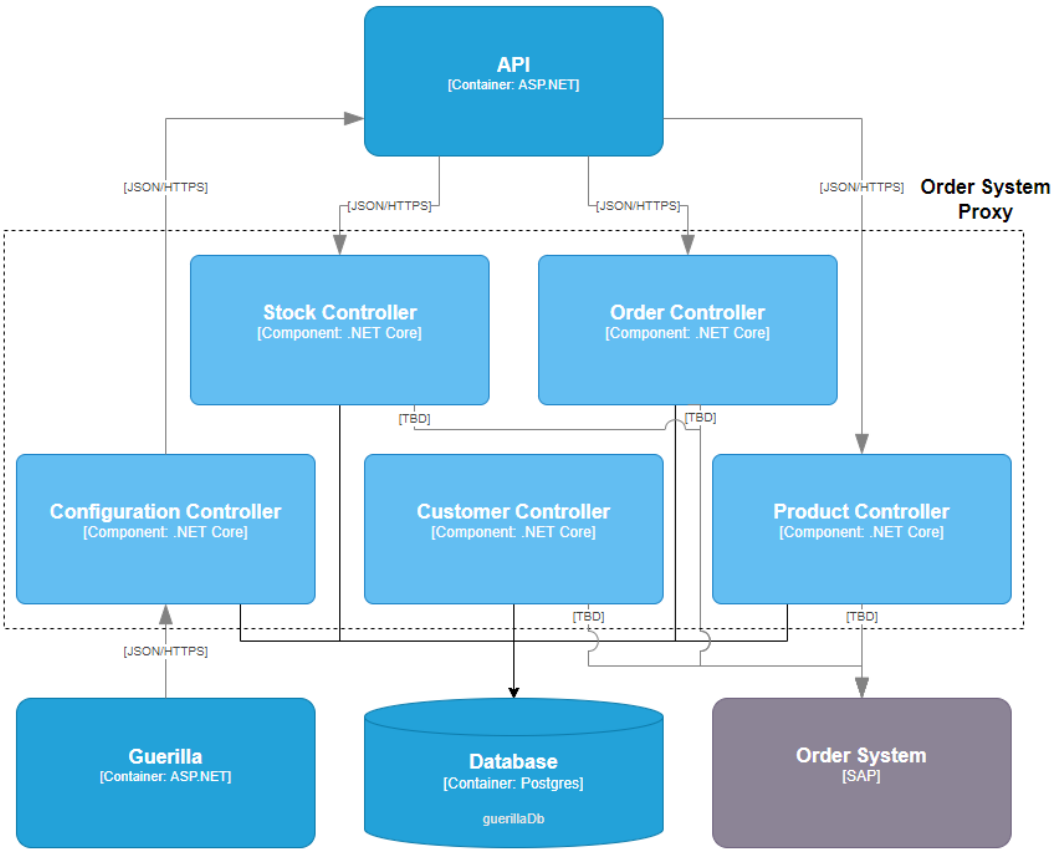

Abbildung 6: Order System Proxy

Der Order System Proxy Container ist dafür zuständig, die Anbindung an ein Bestellsystem (voraussichtlich SAP) zu abstrahieren. Dadurch wird die Kopplung der Bestell-App-Architektur an ein solches System minimiert. Dies ermöglicht einen unkomplizierten Austausch externer Komponenten. Die gewählte Architektur erleichterte es, das SAP im Verlauf der Studienarbeit zu umgehen und mit einer eigenen Implementation (Guerilla Container) zu ersetzen.

Bereich	Technologie
Framework	ASP.NET
Sprache	C#
OR-Mapper	Entity Framework Core
Testing	xUnit.net und Fluent Assertions

Tabelle 33: Order System Proxy Technologien

3.1.6. Production Data Proxy

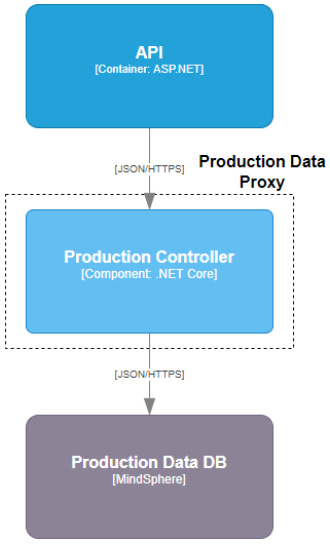

Abbildung 7: Production Data Proxy

Ähnlich wie beim letzten Container besteht die Hauptaufgabe des Production Data Proxy in der Abstrahierung eines externen Systems. In diesem Fall MindSphere, welches der Bestell-App Daten zur Produktion eines Balles liefert.

Bereich	Technologie
Framework	ASP.NET
Sprache	C#
Testing	xUnit.net und Fluent Assertions

Tabelle 34: Production Data Proxy Technologien

3.1.7. Guerilla

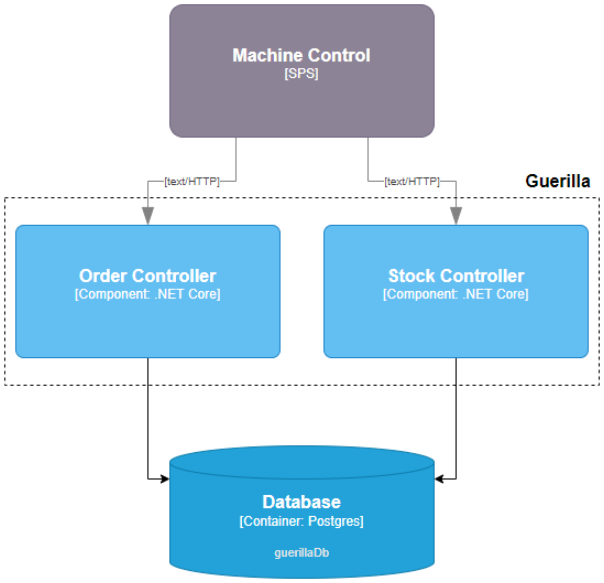

Abbildung 8: Guerilla Container

Als sich im Verlauf der Studienarbeit Verzögerungen der SAP Schnittstellen bemerkbar machten, wurde entschieden, den Guerilla Container zu implementieren. Mit seiner Umsetzung wurde garantiert, dass das Ziel der Arbeit unabhängig der SAP Schnittstellen erreicht werden kann. Durch ihn ist es dem Bestell-App möglich, Bestellungen selber zu verwalten und als Anlaufstelle für die Maschinensteuerung der Smart Factory zu dienen. Der Namen *Guerilla* hat sich ergeben, da eine ungeplante und unkonventionelle Lösung für die Aufgabenerfüllung trotz erschwerenden Einflüssen benötigt wurde.

Bereich	**Technologie**
Framework	ASP.NET
Sprache	C#
Testing	xUnit.net und Fluent Assertions

Tabelle 35: Production Data Proxy Technologien

3.2. Sprachen, Libraries und Frameworks

3.2.1. Mendix

Gemäss Aufgabenstellung wurde die Low-Code Programmiersprache Siemens Mendix[4] evaluiert. Es wurde gegen eine Implementation mit Mendix entschieden. Damit lassen sich keine Web Applikationen, sondern lediglich Mobile Apps entwickeln. Zudem reichen die Kenntnisse der Teammitglieder im Bereich Software Development aus, um nicht auf eine Low-Code Sprache ausweichen zu müssen. Details zur Entscheidung sind im Anhang unter dem Kapitel *Mendix Research* beschrieben.

3.2.2. Datenbank

Es sollen möglichst wenig Informationen (redundant) auf mehreren Systemen gespeichert und das Ordersystem (SAP) als Hauptsystem für Persistenz betrachtet werden. Somit beschränkt sich das Datenmodell hauptsächlich auf die Komponente Benutzerverwaltung. Grundsätzlich müssten sämtliche gespeicherten Informationen basierend auf dem authentifizierten Benutzer bereitgestellt werden. So entspricht das Datenmodell strukturell mehrheitlich einem Baum.

Was spricht für den Einsatz einer NoSQL-Datenbank?

- Die Daten werden in einer ähnlichen Form wie in den Abfragen verwendeten Objekte gespeichert.
- Datenbank lässt sich sehr einfach deployen
- Bietet oft bessere Integration für Echtzeit-Streaming
- Schnellere Abfragen (von Dokumenten)
- Unkomplizierte Migration

Was spricht für den Einsatz einer SQL-Datenbank?

- Datenmigrationen hin zu einem relativem Datenmodell
- Einfache Integration in .NET mit Entity Framework Core möglich
- Erfahrung der Entwickler

Das Datenmodell hätte die Nutzung einer NoSQL-Datenbank zugelassen. Die Anzahl Argumente für den Einsatz einer NoSQL-Datenbank überwiegen klar. Aus Datenbanksicht wäre es die geeignetste Lösung für diese Anwendung gewesen. Aus Entwicklersicht und unter Berücksichtigung der zu erwartenden Abfrage-/Datenmenge können einige Argumente gut relativiert werden. Ausschlaggebend für die Entscheidung zugunsten einer SQL-Datenbank ist die einfache Integration in .NET mittels Entity Framework Core. Für NoSQL-Datenbanken gibt es keine offiziellen Libraries für die Nutzung von Entity Framework Core (Microsoft Azure Cosmos ausgenommen). Dessen Verwendung erleichert die Implementierung von Authentifizierung und genereller Datenbankstruktur (Code-first) ungemein. Natürlich gäbe es ausreichend Anleitungen zur alternativen Integration einer NoSQL-Datenbank oder konkret MongoDB. Risiken und Chancen stehen jedoch nicht mehr in einem gesunden Verhältnis. Damit wurde die Einfachheit der Implementierung höher gewichtet als die generelle Eignung gemäss gegebenem Datenmodell.

Während der Entwicklung wird SQLite verwendet. Eine sehr entwicklerfreundliche Datenbank, da sie keine zusätzlichen lauffähigen Systeme benötigt und sämtliche Daten in einer Datei gespeichert werden. Für die produktive Umgebung wird auf PostgreSQL, die beste relationale Datenbank auf dem Markt, zurückgegriffen.

Die Resultate basieren auf einer Internetrecherche.[5]

3.2.3. Interne Kommunikation

Für die Kommunikation zwischen den einzelnen Containern wurden drei Technologien in Erwägung gezogen: REST Schnittstellen, gRPC und GraphQL. Als Vorbereitung für die spätere Implementierung der Bestell-App wurde eine Kommunikation mit gRPC evaluiert. Aus den daraus gesammelten Erfahrungen wurde entschieden, dass eine Umsetzung mit gRPC keinen Mehrwert gegenüber REST bringen

und lediglich die Komplexität des Technologiestacks erhöhen würde. Ein Vorteil von GraphQL ist, dass nicht zwingend ein komplettes Objekt, sondern auch nur Teile davon ausgetauscht werden können. Dieser Vorteil würde sich dann in der Performance wiederspiegeln. Das Bestell-App wird jedoch keine grosse Datenmenge verarbeiten müssen, weshalb sich das Team für die bereits bekannte Lösung REST entschieden hat.

3.3. Sequenzdiagramm

Es werden die wichtigsten Prozesse der Bestell-App beschrieben. Die Request- und Response-Objekte basieren auf Entitäten im Domain Model und werden durch weitere DTO-Objekte ergänzt.

3.3.1. Benutzer registrieren

Der Benutzer kann sich in der Bestell-App mit der Rolle *User* registrieren.

DTO	Inhalt
ApplicationUser	Der *ApplicationUser* enthält sämtliche Felder zwecks Authentifizierung und Authorisierung.
UserDetails	Zusätzliche Informationen, die für die funktionale Verwendung der Bestell-App benötigt werden, sind in *UserDetails* ausgelagert.
Registration	*Registration* beinhaltet sämtliche benötigte Felder zur Erstellung eines Objektes *ApplicationUser* und dazugehörigen Objekt *UserDetails*.
PublicUser	Beinhaltet veröffentlichbare Informationen aus den Objekten *ApplicationUser* und *UserDetails*

Tabelle 36: DTOs Benutzer registrieren

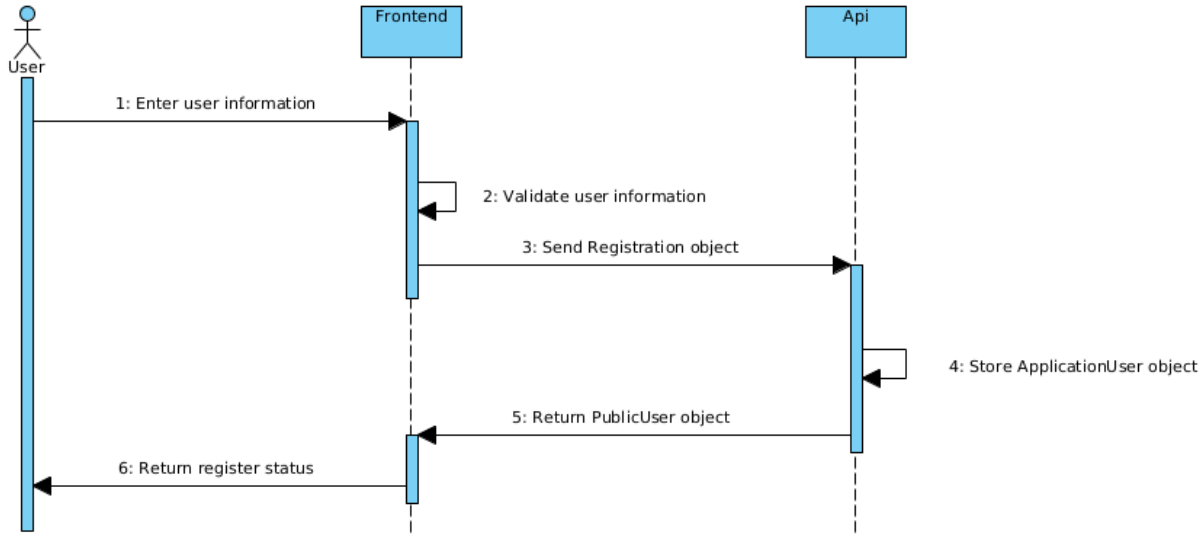

Abbildung 9: Sequenzdiagramm Benutzer registrieren

Schritt	Request	Objekt
3	POST /auth/register	DTO *Registration*
5	-	DTO *PublicUser*

Tabelle 37: Prozessschritte Benutzer registrieren

3.3.2. Kunde registrieren

Ein Kunde wird erstellt, sobald dieser erstmalig für eine Anfrage benötigt wird. Beispielsweise für die Abfrage persönlicher Bestellungen oder bei der Aufgabe der ersten authentifizierten Bestellung.

DTO	Inhalt
Customer	Der *Customer* besteht aus Feldern des Objektes *UserDetails*.
CustomerEntry	Dieses Objekt wird nur für die Kundenregistrierung verwendet. *CustomerEntry* entspricht dem *Customer* Objekt ohne Feld Debitorennummer.

Tabelle 38: DTOs Kunde registrieren

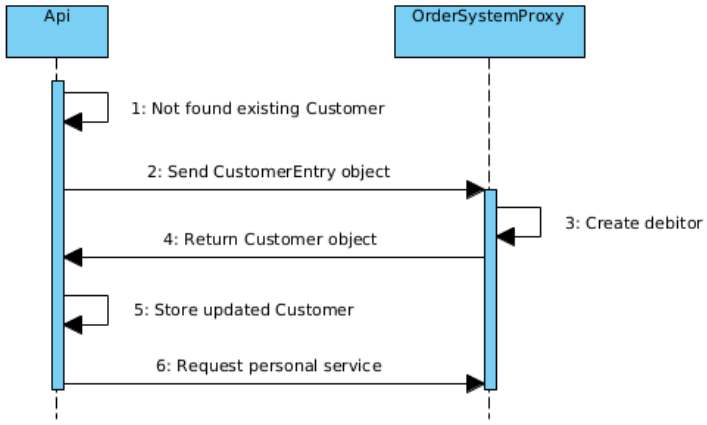

Abbildung 10: Sequenzdiagramm Kunde registrieren

Schritt	Request	Objekt
2	POST /customers	DTO *CustomerEntry* mit Objekt *Customer* ohne Debitorennummer
4	-	Objekt *Customer*

Tabelle 39: Prozessschritte Kunde registrieren

3.3.3. Benutzer anmelden

DTO	Inhalt
LoginCredentials	Beinhaltet die Identitätsidentifikation (Username) und den Identitätsbeweis (Password).
AuthenticatedUser	Beinhaltet das Objekt *PublicUser*, einen Access- und Refresh-Token

Tabelle 40: DTOs Benutzer anmelden

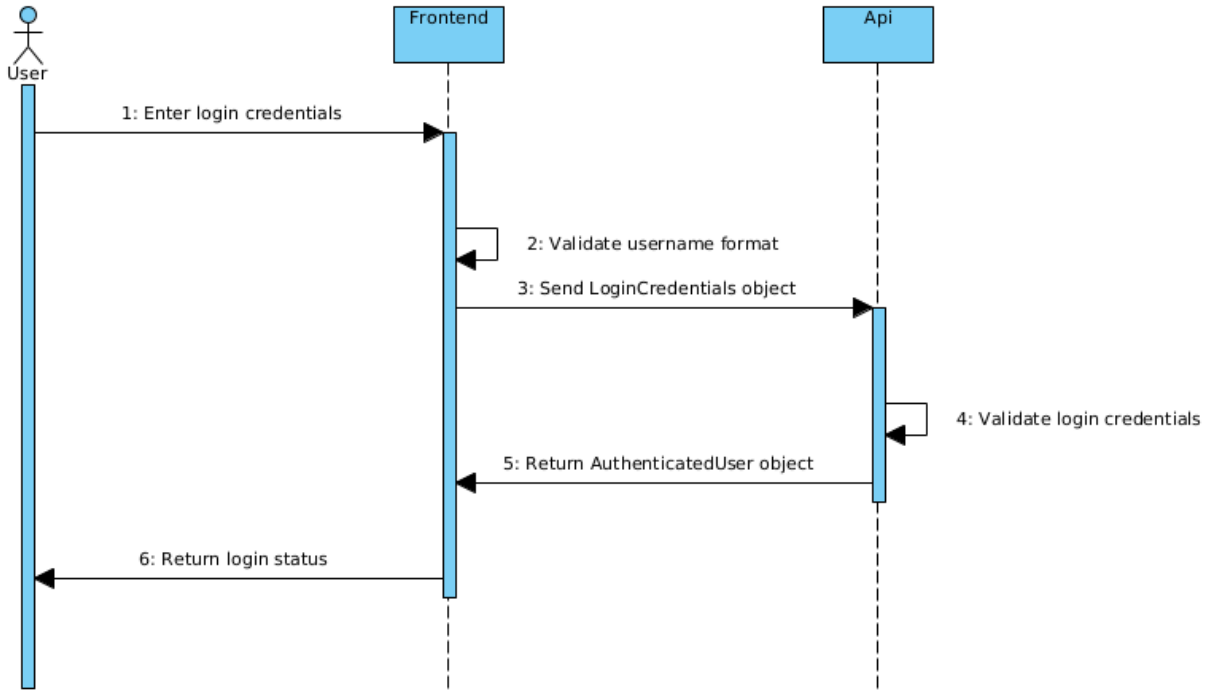

Abbildung 11: Sequenzdiagramm Benutzer anmelden

Schritt	Request	Objekt
3	POST /auth/login	DTO *LoginCredentials*
5	-	DTO *AuthenticatedUser*

Tabelle 41: Prozessschritte Benutzer anmelden

3.3.4. Ball konfigurieren

Damit ein Ball mit verfügbaren Mitteln konfiguriert werden kann, müssen die aktuellen Materialbestände abgefragt werden.

DTO	Inhalt
AbstractBallhalf	Beinhaltet den aktuellen Bestand und die Konfiguration einer Ballhälfte insbesondere dessen Farbe.

StockList	Beinhaltet ein Feld mit einer Liste an Ballhälften und dessen Bestände.

Tabelle 42: DTOs Ball konfigurieren

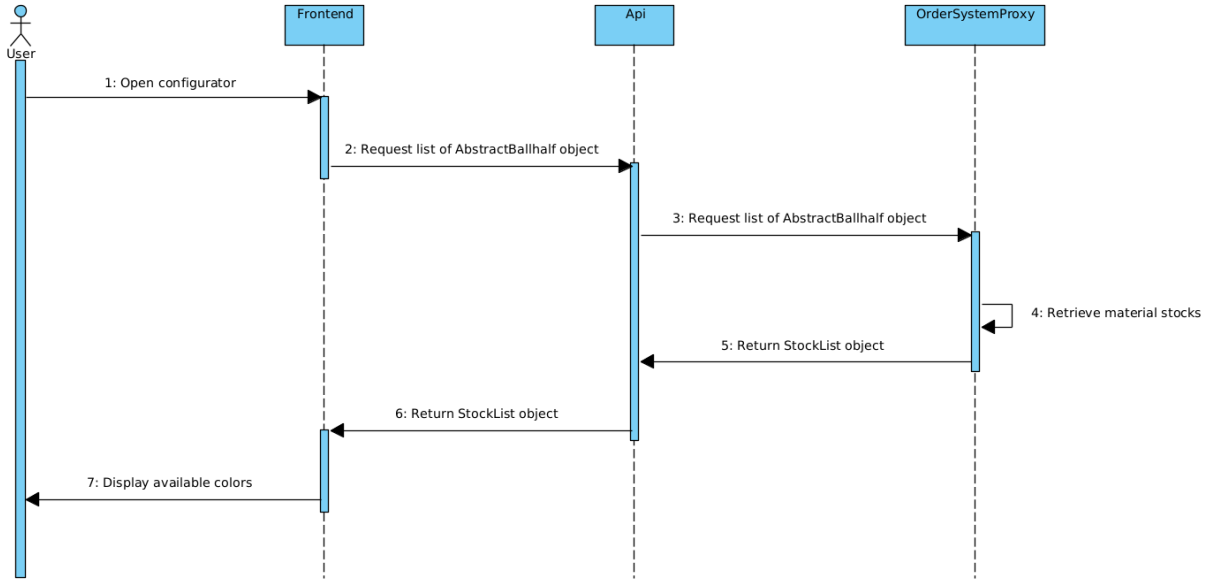

Abbildung 12: Sequenzdiagramm Ball konfigurieren

Schritt	Request	Objekt
2, 3	GET /stocks	Liste mit allen Ballhälften und dessen Bestände.
5, 6	-	DTO *StockList*

Tabelle 43: Prozessschritte Ball konfigurieren

3.3.5. Bestellung aufgeben

Bestellungen können sowohl anonym als auch mit einem authentifizierten Benutzer getätigt werden. Authentifiziert wird über die Headerinformationen der Anfrage.

DTO	Inhalt
FloorballConfiguration	Die *FloorballConfiguration* hält die beiden selektieren Ballhäften in Form der hexadezimalen Farbnummer.
OrderEntry	Die *OrderEntry* besteht aus sämtlichen Feldern, die zum Zeitpunkt der Auftragseingabe bekannt sind (*Customer*, *FloorballConfiguration*)

Tabelle 44: DTOs Bestellung aufgeben

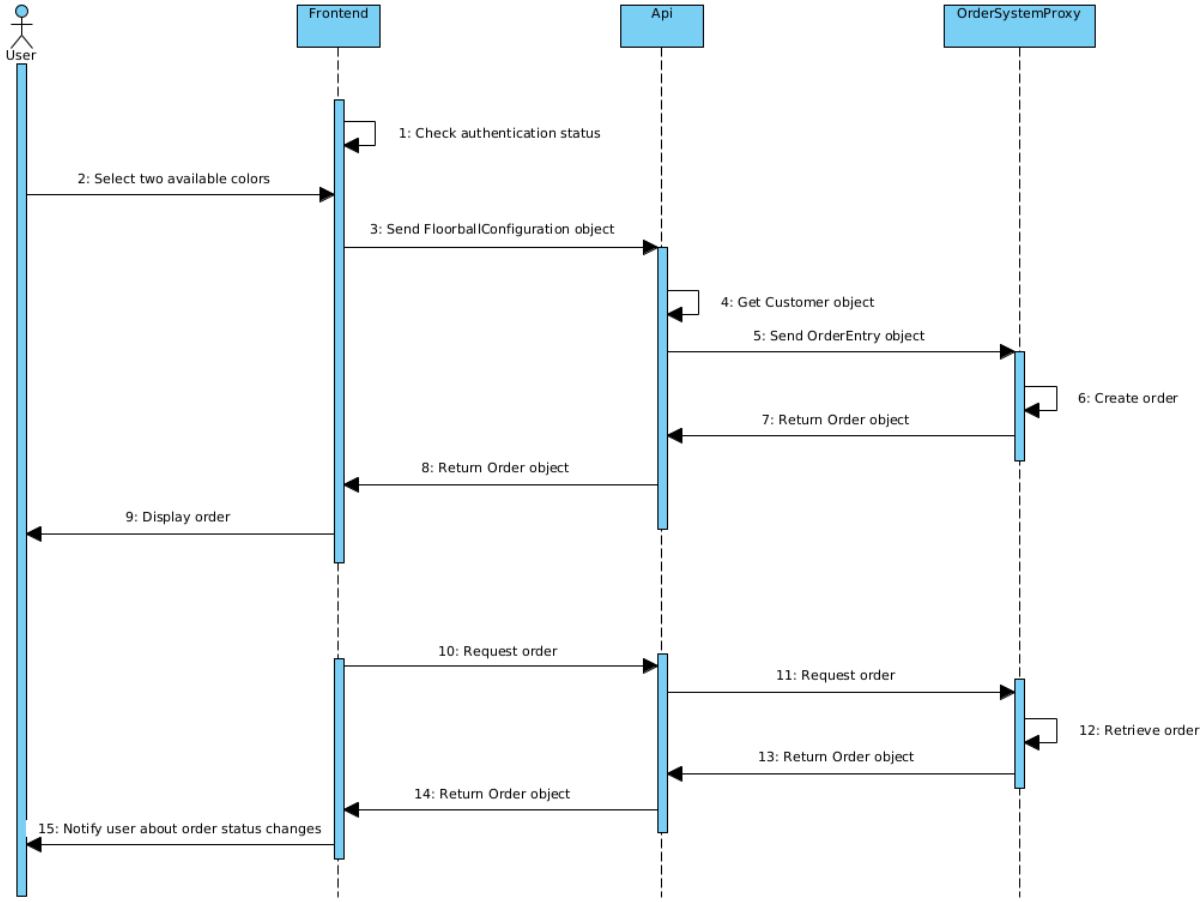

Abbildung 13: Sequenzdiagramm Bestellung aufgeben

Schritt	**Request**	**Objekt**
3	POST /orders	DTO *FloorballConfiguration* mit optionalen Benutzerinformationen im Header
4	-	Erhalte das *Customer*-Objekt des authentifizierten Benutzers oder anonymen Kunden. Bei erster Verwendung wird zusätzlich der Prozess Kunderregistrierung angestossen.
5	POST /orders	DTO *OrderEntry* mit DTO *FloorballConfiguration* und *Customer*
7, 8	-	Objekt *Order* ohne konkrete Ballinformationen
10	GET /orders/{id}	Abfrage einer konkreten Bestellung, mit optionalen Benutzerinformationen im Header
11	GET /orders/{id}	Abfrage einer konkreten Bestellung
13, 14	-	Objekt *Order*

Tabelle 45: Prozessschritte Bestellung aufgeben

3.3.6. Bestellliste abfragen

Bestelllisten können sowohl anonym als auch mit einem authentifizierten Benutzer abgefragt werden. Authentifiziert wird über die Headerinformationen der Anfrage.

DTO	Inhalt
OrderList	Beinhaltet ein Feld mit einer Liste an Bestellungen und dessen Unihockeybällen.

Tabelle 46: DTOs Bestellliste abfragen

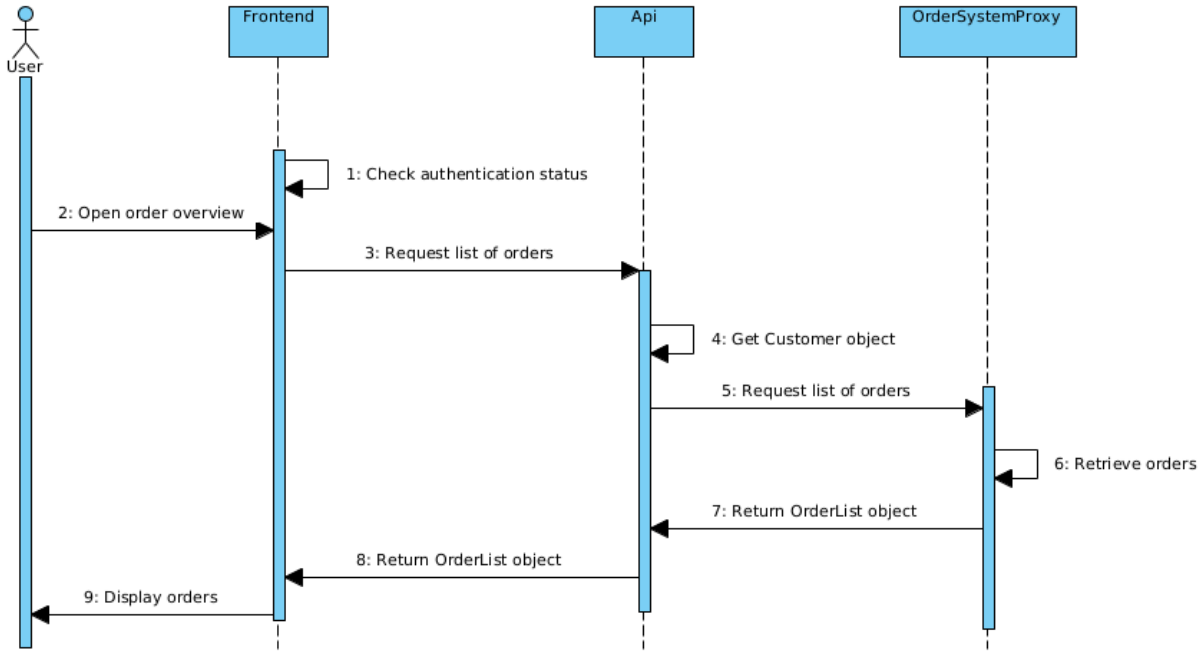

Abbildung 14: Sequenzdiagramm Bestellliste abfragen

Schritt	Request	Objekt
3	GET /orders	Mit optionalen Benutzerinformationen im Header
4	-	Erhalte das *Customer*-Objekt des authentifizierten Benutzers oder anonymen Kunden. Bei erster Verwendung wird zusätzlich der Prozess Kundenregistrierung angestossen.
5	GET /orders	
7, 8	-	DTO *OrderList* ohne konkrete Ballinformationen

Tabelle 47: Prozessschritte Bestellliste abfragen

3.3.7. Produktdaten vom Unihockeyball abfragen

Produktionsdaten eines Balles können mit den dazugehörigen Seriennummern abgefragt werden.

DTO	Inhalt
Floorball	Beschreibt einen echten Unihockeyball mit Ballhälften und Produktionsdaten.
FloorballData	Bündelt *WeldingData* des Unihockeyballs und die *BallhalfData* der beiden Ballhälften. Darin sind jeweilige Produktionsdaten abgelegt.

Tabelle 48: DTOs Produktdaten abfragen

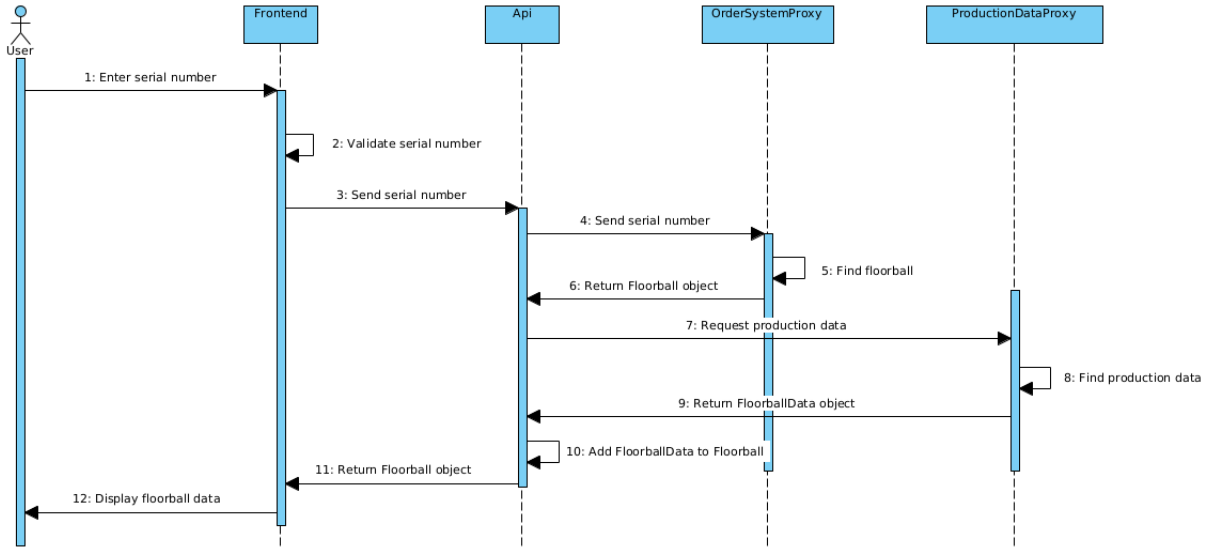

Abbildung 15: Sequenzdiagramm Produktdaten abfragen

Schritt	Request	Objekt
3, 4	GET /products/floorball?...	-
7	GET /production/floorball?...	-
10	-	Verfollständige das Objekt *Floorball* mit den Feldern aus dem Objekt *FloorballData*
11	-	Objekt *Floorball*

Tabelle 49: Prozessschritte Produktdaten abfragen

3.3.8. Konfiguration setzen

Administratoren und Services können die Einstellungen der Bestell-App anpassen.

DTO	Inhalt
Configuration	Beinhaltet Felder zur Definition bestimmter Einstellungen.

Tabelle 50: DTOs Konfiguration setzen

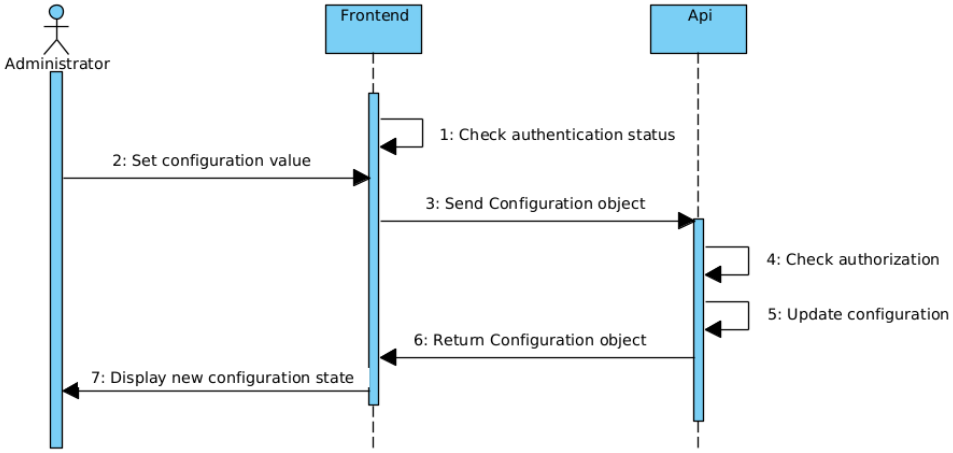

Abbildung 16: Sequenzdiagramm Konfiguration setzen

Schritt	Request	Objekt
3	PUT /configuration/{name}	Passe das *Configuration*-Objekt an.
6	-	Objekt *Configuration*

Tabelle 51: Prozessschritte Konfiguration setzen

3.4. Schnittstellen

3.4.1. Siemens MindSphere

Sämtliche Produktionsdaten der Unihockeybälle können über den sogenannten Fleetmanager von MindSphere abgerufen werden. Für die REST-API stand eine Open-API Dokumentation[6] zur Verfügung. Für die Requests ist eine Authentifizierung nötig, welche mit einem Bearer-Token vorgenommen wird. Eine genauere Beschreibung der API und eine Anleitung, wie dieser Token ausgestellt werden kann, befindet sich im Anhang unter *MindSphere Research*.

3.4.2. SAP

Die SAP Schnittstellen wurden parallel zu der Studienarbeit von einer Studentengruppe der OST St. Gallen erarbeitet. Da der Fokus ihrer Arbeit aber auf der Planung von Prozessen bestand, war es ihnen bis zum Ende dieser Studienarbeit nicht möglich, die nötigen Schnittstellen bereitzustellen. Dieser Umstand verunmöglichte eine Kommunikation mit der SAP Instanz der OST. Aus diesem Grund wurde eine projektinterne Bestellungsverwaltung und eine direkte Anbindung an die Maschinensteuerung implementiert.

3.5. Deployment

Das Continuous Deployment während der Entwicklung wird auf dem Testserver folgendermassen aufgebaut.

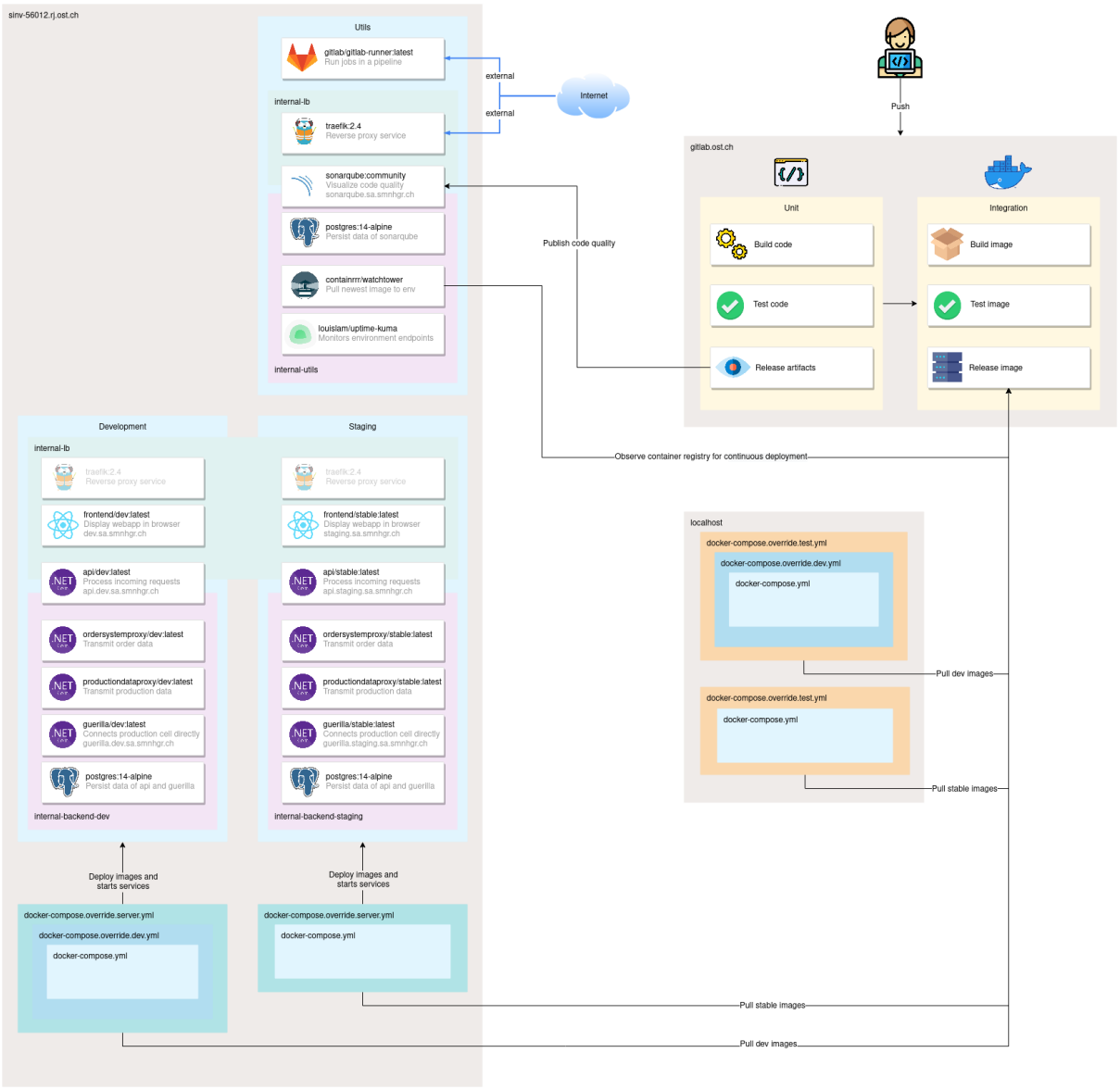

Abbildung 17: Development Deployment Übersicht

Das produktive Environment kann dagegen stark vereinfacht gestartet werden. Alle Anleitungen zum Deployment von Entwicklungs- und Produktivumgebungen sind in entsprechenden Repository READMEs beschrieben.

3.5.1. Branches

Jedes Projekt hat zwei beständige Branches *Development* und *Master*. Entwickelt wird in weiteren Branches, welche auf ein Issue (Task) referenzieren.

Branch	Beschreibung
master	Der Code in diesem Branch ist stabil und kann jederzeit für den produktiven Einsatz verwendet werden. Auch die Interoperabilität mit anderen Systemkomponenten und Umsystemen ist gewährleistet. Der Entwicklungsstand enthält nahezu keine Erros und Defects mehr.
development	Enthält den zusammengetragenen Code aus den verschiedenen Issues. Dieser ist für sich lauffähig und kann im Gesamtsystem eingesetzt werden. Schwerwiegende Errors und Defects sind behoben. Werden die Qualitätsansprüche vom *master* Branch erfüllt, soll der aktuelle Commit in den *master* Branch verschoben werden.
$ISSUE_ID-$ISSUE_NAME	Diese Branches werden für die tägliche Weiterentwicklung des Systems verwendet. Sie sind in der Regel nicht stabil und erhalten regelmässig Aktualisierungen. Werden die Qualitätsansprüche vom *development* Branch erfüllt, soll der aktuelle Commit in den *development* Branch verschoben werden.

Tabelle 52: Continuous Delivery/Deployment (CD)

3.5.2. Infrastruktur

Die Bestell-App wird in folgenden Systemen entwickelt und getestet.

System	Bezug/Zweck
localhost	Die Applikation wird lokal weiterentwickelt und getestet.
gitlab.ost.ch	Eine GitLab-Instanz der OST, worin der gesamte Prozess Softwareentwicklung vom Commit-Push bis zum Image-Release abgewickelt wird. Zusätzlich wird darin das Projektmanagement betrieben und der Product Backlog geführt.
sinv-56012.rj.ost.ch	Ein virtuelle Maschine, worauf alle eigenen Tools sowie das Deployment mit den Umgebungen *Development* und *Staging* laufen.

Tabelle 53: Infrastruktur

Für die erfolgreiche Entwicklung und Distribution der Softwarelösung spielen folgende Tools eine zentrale Rolle.

Tool	Bezug/Zweck
Docker	Docker vereinfacht die Erstellung von Images beziehungsweise Bereitstellung von einzelnen Containern, indem es alle nötigen Pakete einer Systemkomponente in einem Image bündelt und es vom Umsystem (OS) unabhängig lauffähig macht. Für jedes in der Containerübersicht dargestellte Systemkomponente wird ein eigenes Image erstellt und dessen Dienstleistung für das Gesamtsystem in einem separaten Container bereitgestellt.

Docker Compose	Mithilfe von Compose lassen sich Multi-Container-Docker-Applikationen in einer YAML-Datei definieren und mit einem einzigen Befehl kontrolliert ausführen. Sowohl die Environments als auch das gesamte produktive Bestell-App-System wird in einer Compose-Datei beschrieben.
Traefik	Traefik ist ein Reverse Proxy und Loadbalancer. Dieser nimmt Anfragen an das System entgegen und leitet sie an für die Bearbeitung zuständige Container weiter. Zusätzlich setzt der Reverse Proxy konsequent HTTPS/TLS durch und kümmert sich um die Setzung von Headerinformationen wie CORS.
SonarQube	SonarQube führt statische Analysen durch und bewertet die technische Qualität von Code. Dafür analysiert es den Code auf verschiedene Qualitätsmerkmale und bereitet die Resultate visuell auf.
Watchtower	Watchtower von Containrrr überwacht die Container Registry im GitLab auf Änderungen. Wird ein neueres Image gefunden, lädt Watchtower es herunter, stoppt die bestehenden Container und startet diese mit den neuen Image aber bestehenden Einstellungen wieder neu. Somit kann die laufende Image-Version in den Environments einfach aktualisiert und Continuous Deployment ermöglicht werden.
Uptime	Uptime Kuma ist ein sehr schlankes Monitoring Tool für die Überwachung der Verfügbarkeiten externer Schnittstellen und eigener Umgebung. Es bietet eine übersichtliche Statusseite mit sämtlichen registrierten Endpunkten.

Tabelle 54: Tooling/Utils

3.5.3. Continuous Integration (CI)

Jeder hochgeladene Commit durchläuft mehr oder weniger Schritte

Schritt	**Branches**	**Beschreibung**
Build code	alle	Native Code wird generiert und für die Tests vorbereitet.
Test code	alle	Unit Tests und Integration Tests eines einzelnen Containers werden durchgeführt. Der Container wird in sich vollständig getestet. *Development* und *Staging* laufen.
Release artifacts	Development, Master	Build- und Testresultate beziehungsweise deren Artifakte werden selektiert und für Umsysteme bereitgestellt. Code und Code Test Reports werden an SonarQube weitergeleitet und ist unter der URL https://sonarqube.{domain} erreichbar. OpenAPI-Schnittstellenbeschreibungen werden veröffentlicht.
Build image	Development, Master	Ein Dockerfile listet die Instruktionen zum Bau und Betrieb des Containers auf. Docker generiert daraus mithilfe des Docker-Daemon ein Image.
Test image	Development, Master	Das gebaute Image durchläuft Integration Tests.
Release image	Development, Master	Erfolgreich erstellte und getestete Images werden in der Container Registry vom OST GitLab abgelegt.

Tabelle 55: Continuous Integration (CI)

3.5.4. Continuous Deployment (CD)

Jeder in der Architektur beschriebene Container wird nach erfolgreicher CI-Pipeline als Docker-Image veröffentlicht. Auf diese Images kann mittels eines Deploy Tokens zugegriffen werden. Für die Entwicklungsphase bedient sich der Server sinv-56012.rj.ost.ch dieser Möglichkeit, um die bereitgestellten Images für die beschriebenen Environments herunterzuladen. Alternativ kann für das lokale Testen auch das klassiche GitLab-Login der OST mit passenden Gruppenrechten für das Image-Pulling verwendet werden.

Für das Deployment stehen unter registry.gitlab.ost.ch:45023/sa_unihockey-ball/$PROJECT_NAME/ folgende Image-Versionen zur Verfügung.

Pfad (Tag)	Source Branch	Beschreibung
stable:latest	master	Dieses Image entspricht immer dem zuletzt erfolgreich gebauten Image basierend auf dem Master-Branch.
stable:v$PIPELINE_IID	master	Jedes erfolgreich gebaute Image auf dem Master-Branch wird mit einer inkrementellen Versionsnummer abgelegt.
dev:latest	development	Damit steht der letzte, erfolgreiche Image-Build vom Development-Branch zur Verfügung.
test:...	alle	Diese Images werden für (Continuous Integration-)Tests verwendet. Diese können Errors enthalten, stehen nur temporär zur Verfügung und werden mittelfristig automatisch gelöscht.

Tabelle 56: Images in der Container Registry

Diese Images finden in verschiedenen Environments ihre Verwendung. Die meisten dienen der kontinuierlichen Integrations- und Interoperabilitätsprüfung von neuem Code im Gesamtsystem sowie im Zusammenspiel mit der Deployment-Konfiguration in einer produktivnahen Umgebung. Für den Kunden liefern Environments einen Einblick in den laufenden Softwarentwicklungsprozess.

Umgebung	Beschreibung
Utils	Dieses Environment enthält sämtliche Systeme ausserhalb von GitLab, die für eine funktionierende, qualitätssichernde CI/CD Pipeline notwendig sind. Beispielsweise sind SonarQube und Watchtower hier angesiedelt. Im gesamten Testumfeld wird der Reverse Proxy von dieser Umgebung verwendet, da mehrere Reverse Proxy-Instanzen sich gegenseitig stören. Diese sind jedoch equivalent konfiguriert.
Test	Dieses Setup wird für (Continuous Integration-)Tests und die lokale Entwicklung eingesetzt. Es liefert dem Entwickler ein lokal lauffähiges und zugleich offenes Environment zur Integration seiner Änderungen ins bestehende System.
Development	Enthält die Images vom *development* Branch. Ansonsten unterscheidet sich dieses kaum von der produktiven Konfiguration. Es ermöglicht eine frühzeitige Erkennung von Fehlern zwischen Containern.
Staging	Dieses Environment entspricht der produktiven Umgebung. Es verwendet die letzten, stabilen Image-Release vom *master* Branch. Während der Entwicklung sprechen wir von *Staging*. Sobald diese an den Kunden übergeben wird und in dessen Umgebung läuft von *Production*. In diesem Environment werden Usability Tests durchgeführt und Feedback vom Kunden eingeholt.

Production	Dieses Environment läuft produktiv beim Kunden (aktuell nicht umgesetzt). Es unterscheidet sich von *Staging* lediglich in einigen Environment Variablen, welche statt auf Test- nun auf produktive Umsysteme zeigen.

Tabelle 57: Environments

Die Environments werden in verschiedenen Compose-Dateien spezifiziert. Dabei dient immer das produktive System als Ausgangslage und wird individuell angepasst beziehungsweise überschrieben.

Konfigurationsdatei	Beschreibung
docker-compose.yml	Enthält die gesamte Container-Konfiguration für den produktiven Einsatz der Bestell-App. Wird auch für das *Staging* Environment verwendet.
docker-compose.override.dev.yml	Nutzt die produktive Konfiguration und ersetzt dessen Images mit jenen vom *development* Branch.
docker-compose.override.server.yml	Ergänzt die bestehende Konfiguration mit zusätzlichen Werten für das Continuous Deployment und den Einsatz auf dem Testsystem. Es deaktiviert überzählige Reverse Proxies.
docker-compose.override.test.yml	Passt die Konfiguration für den Einsatz auf localhost an und stellt zusätzliche Tools zur Entwicklung bereit (Bsp. visuelles Datenbank-Interface). Als Ausgangslage kann das *Development* Environment oder das *Staging* Environment dienen.

Tabelle 58: Konfigurationsdateien

3.6. Wireframes

Anhand der erarbeiteten Use Cases werden in diesem Kapitel die Wireframes dargestellt, welche die Struktur der Bestell-App visualisieren. Als Inspiration wurden die Wireframes aus der Bachelorarbeit von Robin Züllig verwendet. Diese wurden aber alle gemäss den aktuellen Anforderungen überarbeitet. Zusätzlich wurden Desktop-Ansichten ergänzt. Das Design der Bestell-App wird hier nicht festgehalten, der Fokus liegt lediglich auf dem Aufbau. In der Umsetzung werden jedoch Farben und Schriftarten dem OST-Design entsprechen.

3.6.1. Desktop

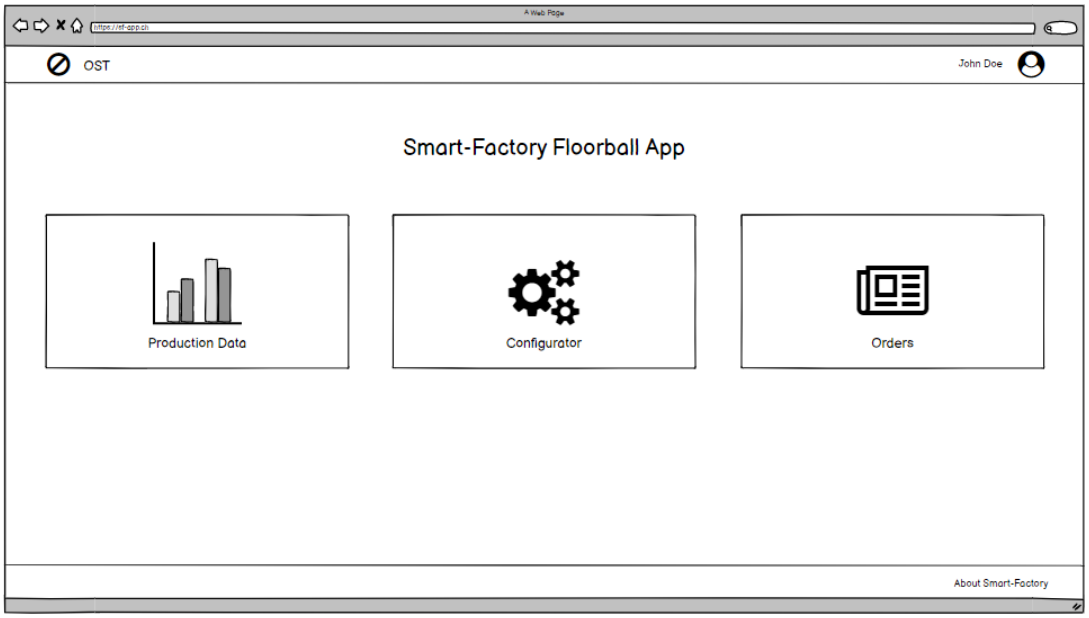

Abbildung 18: Desktop - Home

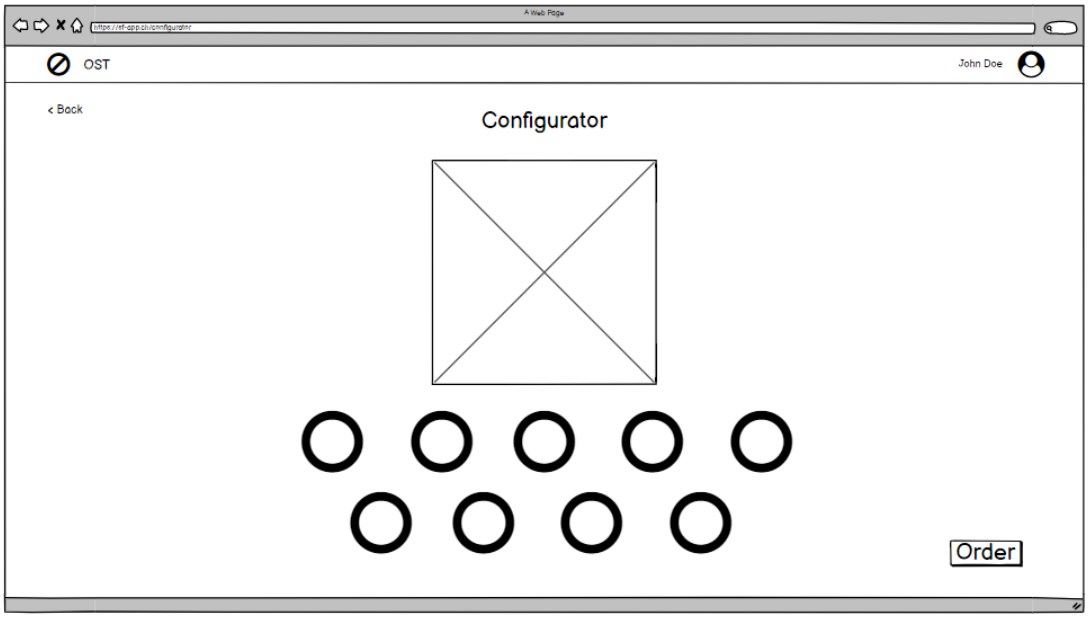

Abbildung 19: Desktop - Konfigurator v1

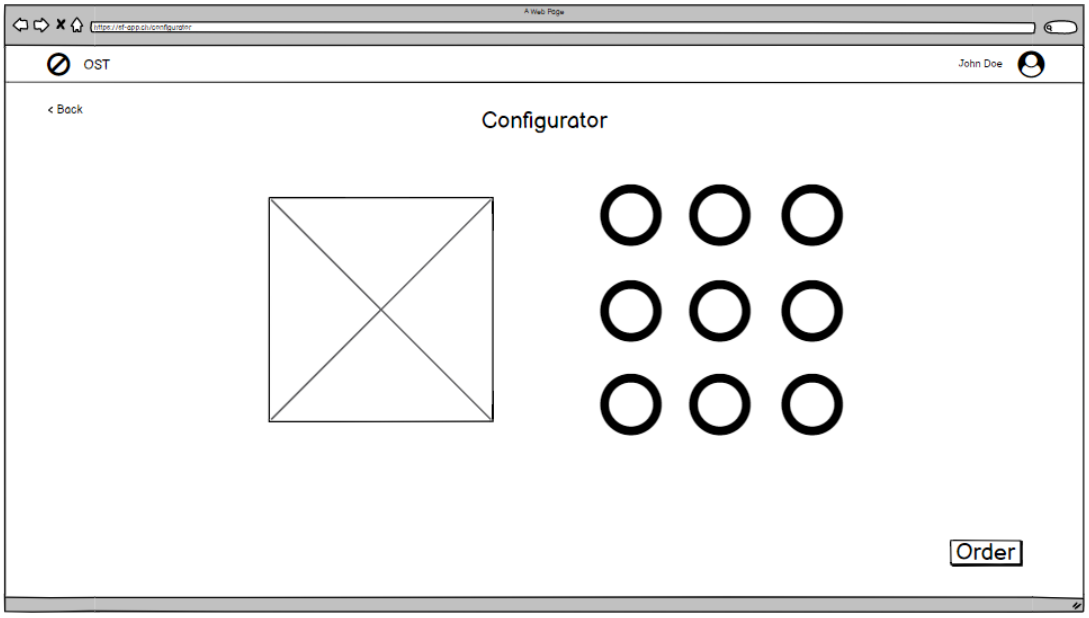

Abbildung 20: Desktop - Konfigurator v2

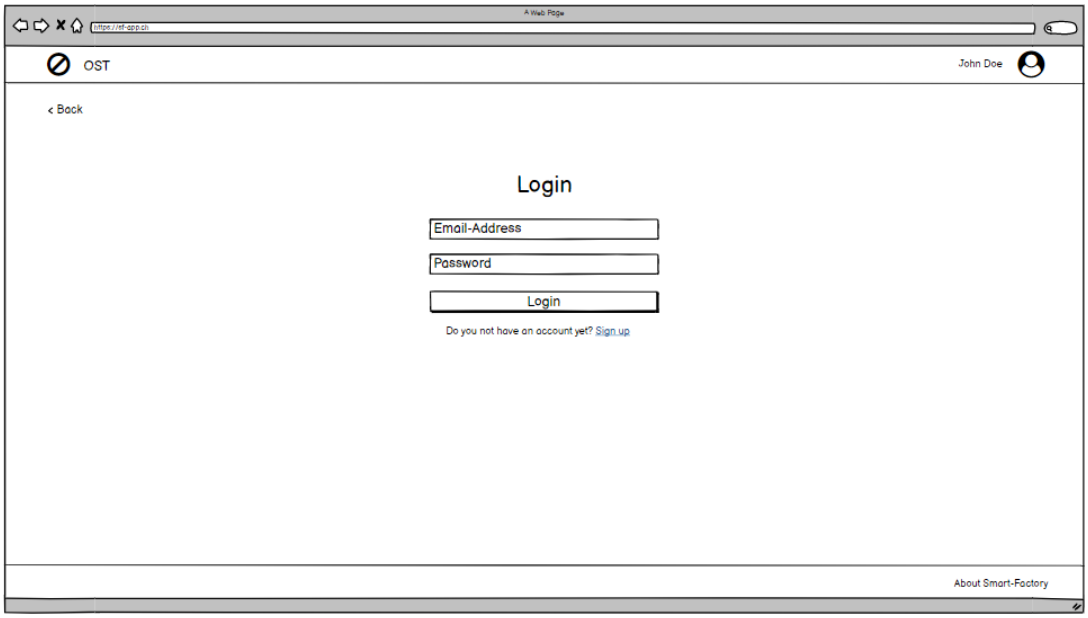

Abbildung 21: Desktop - Login

Abbildung 22: Desktop - Registrierung

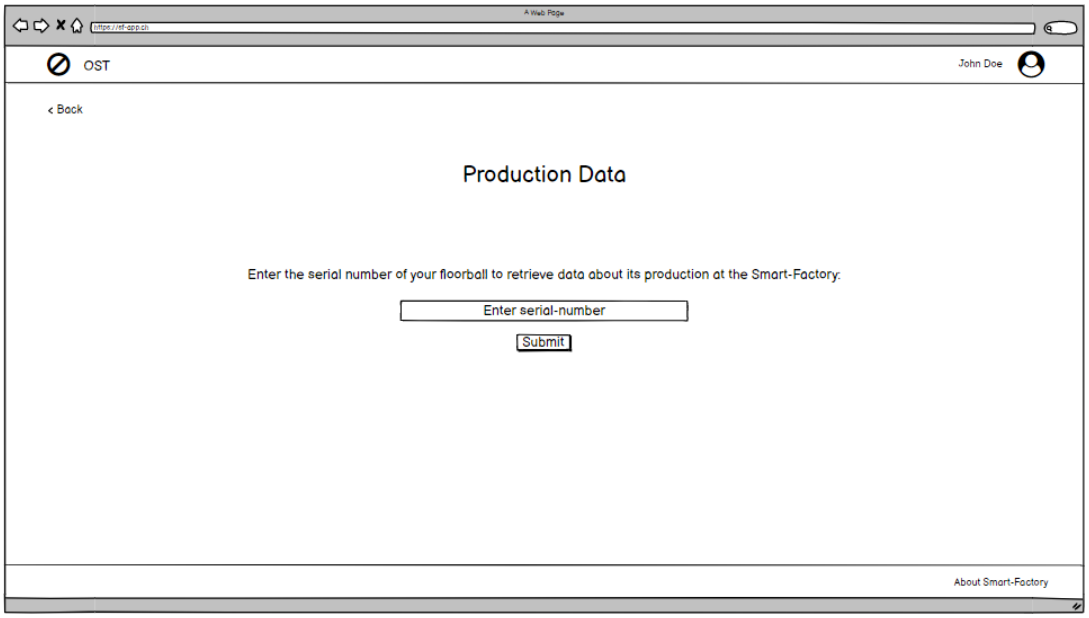

Abbildung 23: Desktop - Production Data 1

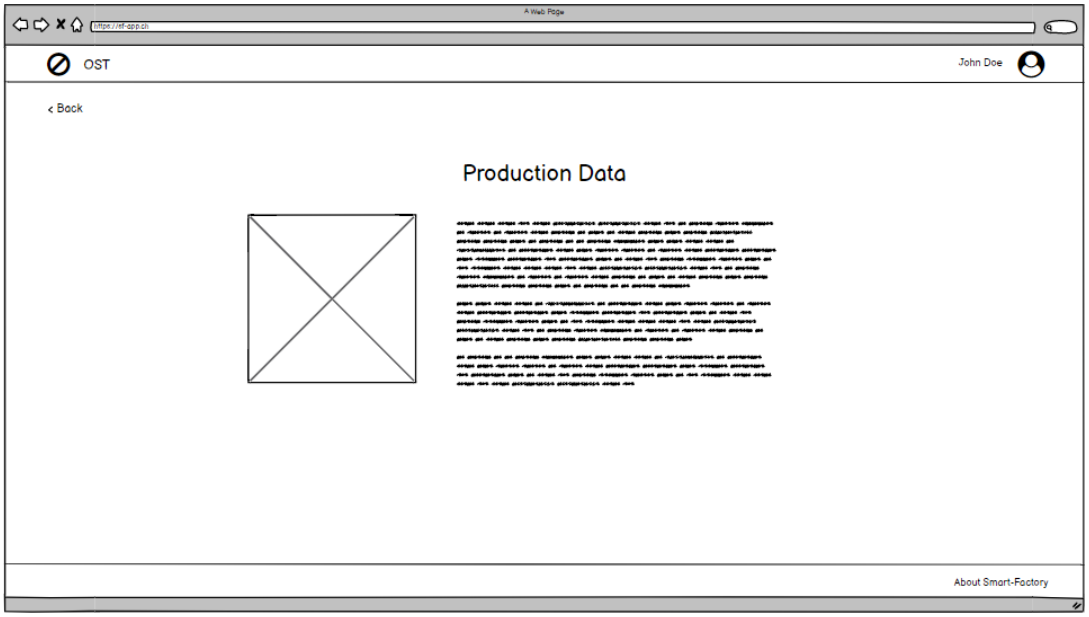

Abbildung 24: Desktop - Production Data 2

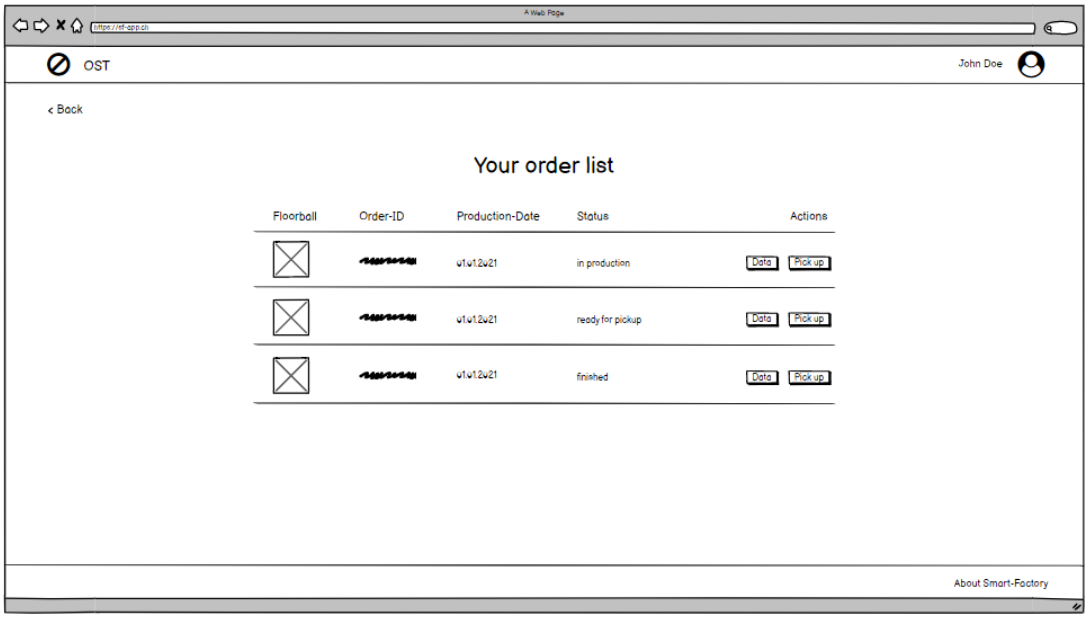

Abbildung 25: Desktop - Bestellliste

Abbildung 26: Desktop - Bestellung abholen

3.6.2. Mobile

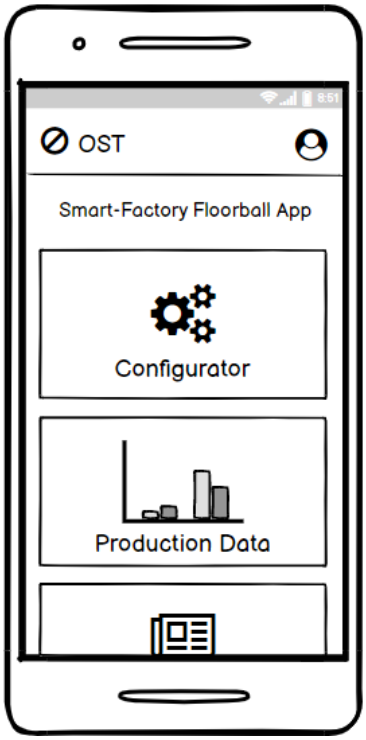

Abbildung 27: Mobile - Home

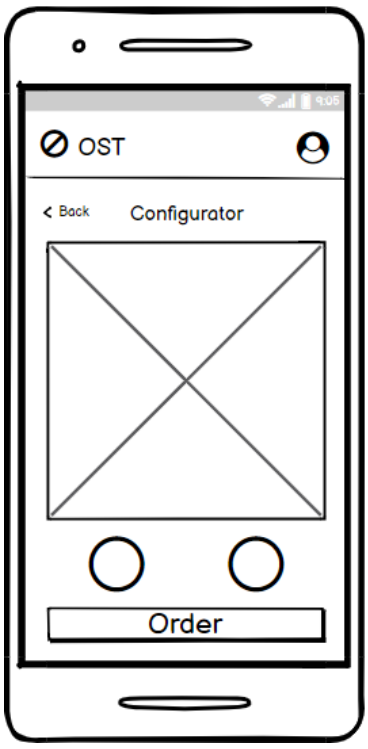

Abbildung 28: Mobile - Konfigurator

Abbildung 29: Mobile - Login

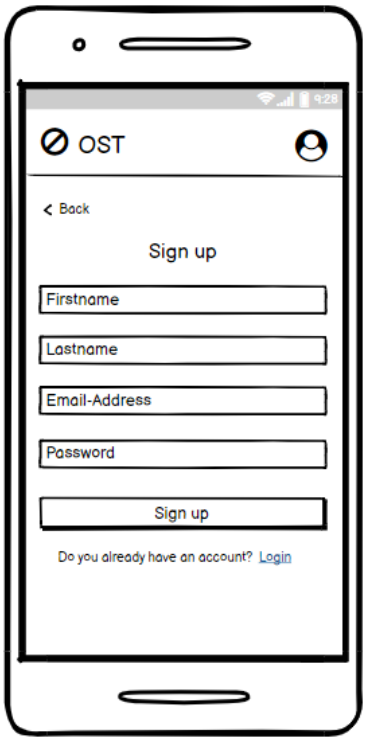

Abbildung 30: Mobile - Registrierung

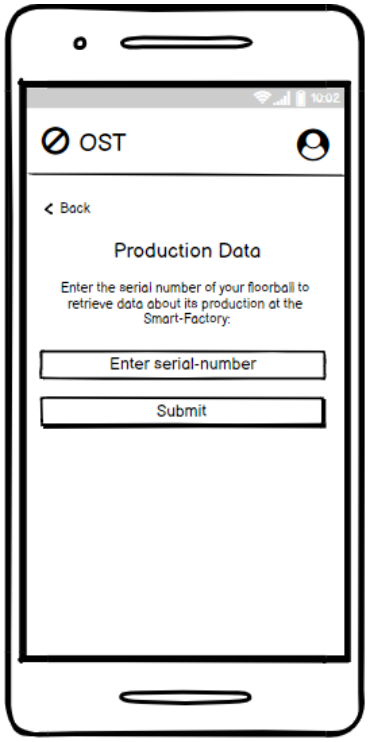

Abbildung 31: Mobile - Production Data 1

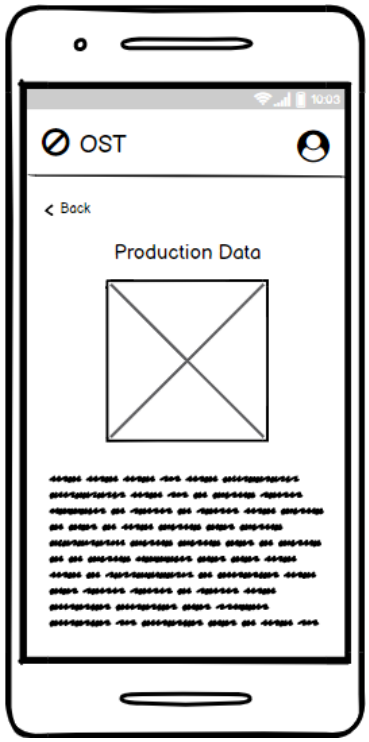

Abbildung 32: Mobile - Production Data 2

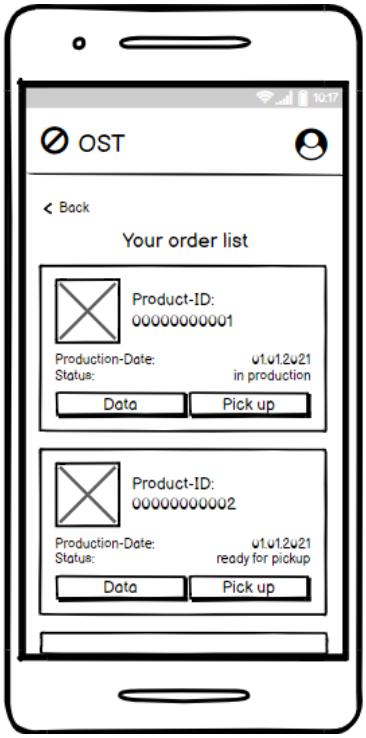

Abbildung 33: Mobile - Bestellliste

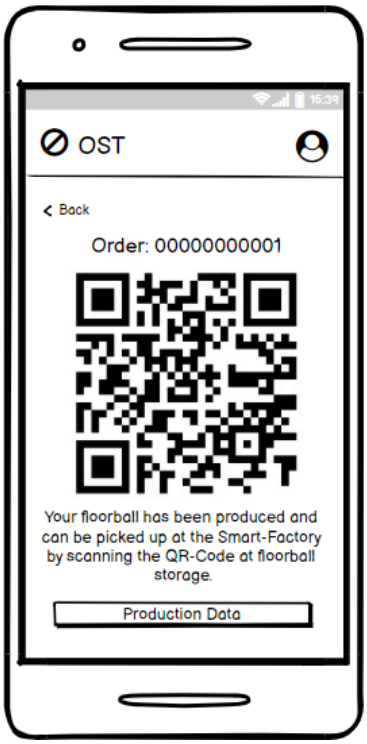

Abbildung 34: Mobile - Bestellung abholen

4. Implementierung

4.1. Projektstruktur

Das Projekt ist in insgesamt zehn Repositories unterteilt. Diese liegen auf GitLab Instanz der OST. Die konkrete Nutzung der einzelnen Repositories ist im entsprechenden README detailierter beschrieben.

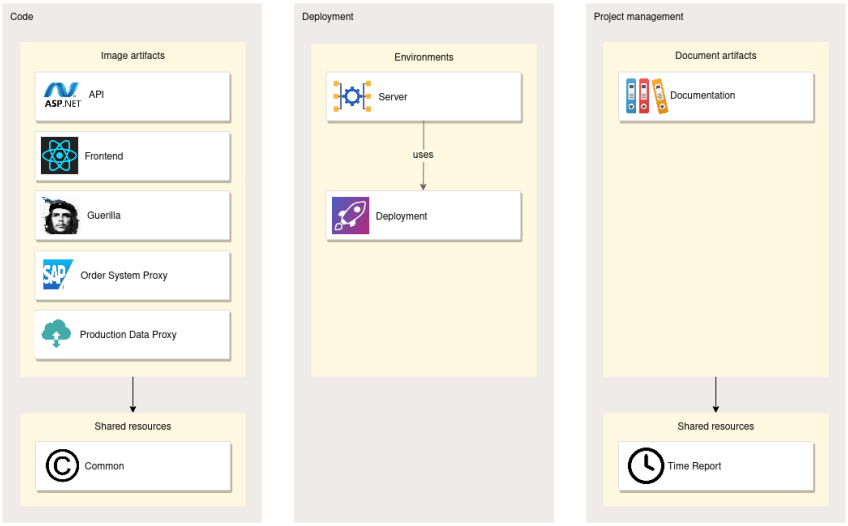

Abbildung 35: Projektstruktur

4.1.1. Code

Die Bestell-App nutzt bis zu fünf selbstentwickelte Container. Jeder dieser Container entspricht einem eigenen Projekt (in C# einer Solution) und ist in einem separaten Repository abgelegt. Als Resultat wird ein Image generiert, womit der Container gestartet und im Environment genutzt werden kann.

In diesen Repositories werden folgende Aufgaben erledigt:

- Code Building
- Unit Testing
- Quality control
- Artifacts Generating
- Image Building

Repository	Beschreibung
API	ASP.NET Core Solution
Frontend	React App
Guerilla	ASP.NET Core Solution
Order System Proxy	ASP.NET Core Solution
Common	Das Common Repository hält gemeinsam genutzte Definitionen und Daten für produktive Assemblies und die Tests. Dieses ist als Submodule in die anderen Code-Repositories eingebunden.

Tabelle 59: Konfigurationsdateien

4.1.2. Deployment

Für das Deployment wird zwischen zwei Repositories unterschieden.

Repository	Beschreibung
Deployment	In diesem Repository ist die gesamte Konfiguration für die Environments *Production*, *Development* und *Testing* abgelegt. Damit lässt sich die Bestell-App veröffentlichen. Zusätzlich beinhaltet dieses Repository CI/CD Konfigurationsvorlagen für Code Repositories. Damit wird eine möglichst einheitliche Pipeline garantiert.
Server	Damit lassen sich zwei Environments parallel auf dem gleichen Host betreiben und zusätzliche Tools für die Entwicklung starten. Als Ausgangslage dient das Repository *Deployment*, welches als Submodule eingebunden ist. Die aktuelle Konfiguration ist für den Betrieb zuhanden der Studienarbeit ausgelegt.

Tabelle 60: Konfigurationsdateien

4.1.3. Projekt management

Die Dokumentation und Zeiterfassung sind ebenfalls versioniert.

Repository	Beschreibung
Documentation	Der Projektfortschritt ist in diesem Repository dokumentiert. Die Projektdokumentation ist in LaTeX geschrieben und steht als Artifakt im PDF-Format zur Verfügung.
Time Report	Analysiert die Issues von GitLab und generiert täglich um 2 Uhr Auswertungen zur geleisteten Arbeit. Die Ergebnisse stehen als Artefakt im PNG-Format zur Verfügung.

Tabelle 61: Konfigurationsdateien

4.2. Qualitätsnachweis

4.2.1. Code

In allen Projekten der Studienarbeit werden Linter eingesetzt, welche den Code statisch analysieren und vor Programmierfehlern, Bugs, stilistischen Fehlern und verdächtigen Konstrukten warnen. Zusätzlich werden Code-Formatierer verwendet um einen einheitlichen Code Style zu erzwingen. Diese Tools sind eine grosse Unterstützung, um einen hohen Standard der Code Qualität einzuhalten. Folgende Produkte wurden in den verschiedenen Projekten verwendet.

	Frontend (TypeScript)	**Backend (C#)**
Linter	ESLint[7]	SonarLint Analyzer[8]
Formatter	Prettier[9]	StyleCop[10]
Testing framework	Testing Library mit Jest[11]	xUnit.net[12] FluentAssertions[13]
Mocking	-	Moq[14]

Tabelle 62: Testing Tools

Die automatisierten Tests im Frontend bestehen aus Unit-, Komponenten- und Integrations-Tests. Erstere werden verwendet, um die Qualität der Hilfsfunktionen zu gewährleisten. React Komponenten werden mit der Hilfe von Jest als solche getestet. Diese Tests beschränken sich aber bei komplexen Komponenten darauf, ob sie dargestellt werden können oder nicht. Aus diesem Grund wird ihr Verhalten bei Benutzerinteraktionen als Teil der Systemtests manuell überprüft. Detailierter werden Redux, beziehungsweise die Reducer und Actions getestet. Damit wird gewährleistet, dass Zustandsänderungen der Bestell-App wie gewünscht verlaufen und keine Seiteneffekte mit sich bringen.

Innerhalb der Backend-Projekte wird zwischen Unit- und Integration-Tests unterschieden. Unit-Tests testen nur eine einzelne Klasse. Integration-Tests dagegen testen das Zusammenspiel mehrerer Klassen und die Reaktion des ganzen Containers auf erwartetbare Anfragen. Externe Abhängigkeiten werden dafür durch Mocks simuliert. Das Mock-Seeding und Migrationen wird mehrheitlich in den Tests ausgeklammert. Komponenten mit Datenbankzugriffen wurden trotz einigen Schwierigkeiten bestmöglich getestet.

Die Resultate aus den statischen Analysen und Tests werden kontinuierlich auf die Plattform SonarQube hochgeladen und visuell aufbereitet dargestellt.

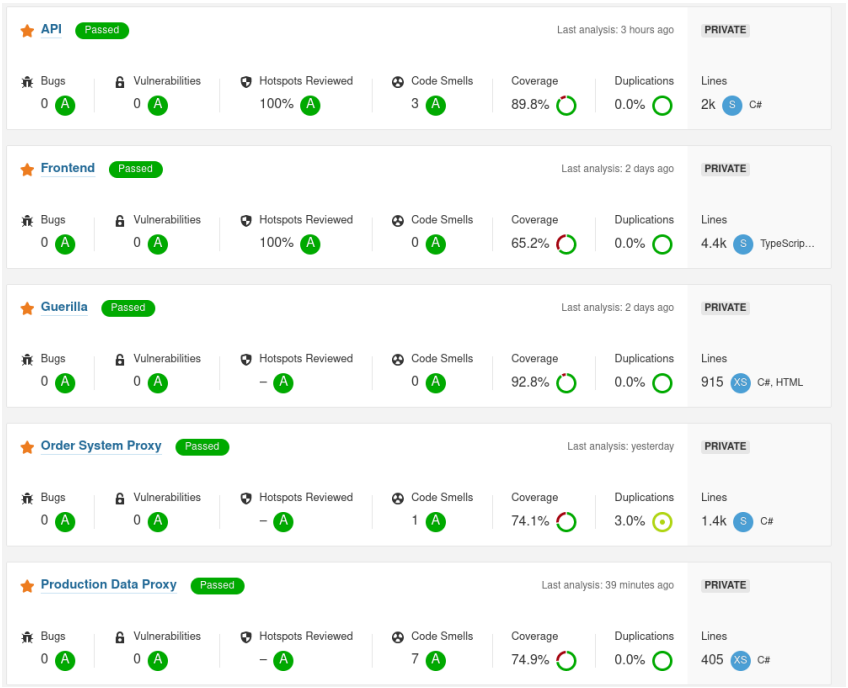

Abbildung 36: SonarQube Resultate

Die Resultate und Einstellungen stehen online unter https://sonarqube.{domain} zur Verfügung.

4.2.2. Google Lighthouse

Die Qualität des Frontends wurde unter anderem anhand der automatisierten Google Lighthouse Checks überprüft und verbessert. Das Tool durchläuft eine Bandbreite an Tests in den Bereichen Performance, Accessibility und SEO (Search Engine Optimization) an einer gewünschten Webseite. Daraus wird ein Rapport über deren Qualität generiert. Das Tool wurde auf der Bestell-App ausgeführt und die fehlgeschlagenen Tests anschliessend dazu verwendet, die App zu verbessern. Nach einigen Optimierungen wurde letztendlich folgendes Resultat erreicht:

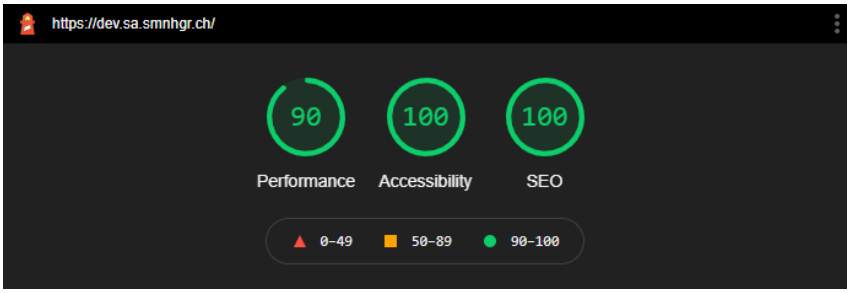

Abbildung 37: Google Lighthouse Resultate

4.2.3. Responsive Design

Die Bestell-App wird voraussichtlich in erster Linie auf mobilen Geräten verwendet. Daher ist ein Responsive Design ein wichtiges Qualitätsmerkmal. Um diesem gerecht zu werden, wurde die App auf folgenden Bildschirmgrössen simuliert, getestet und für die üblichsten Grössen optimiert:

- 2560x1440
- 1920x1080
- 1680x1050
- 1440x900
- 1280x800
- 1194x834 (iPad Pro 11)
- 428x926 (iPhone 12 Pro Max)
- 393x851 (Google Pixel 5)
- 390x844 (iPhone 12 Pro)
- 360x800 (Samsung Galaxy S21 Ultra)
- 360x780 (Huawei P30 PRO / iPhone 12 Mini)

Responsive Screenshots der wesentlichen Seiten sind im Anhang zu finden.

4.2.4. Systemtests

Für Tests des vollen Funktionsumfangs der Bestell-App wurde eine Checkliste entwickelt. Ihr Fokus liegt auf dem Zusammenspiel der verschiedenen Softwarekomponenten. Dies erfolgte über diverse Aktionen im Frontend, welche Abläufe im gesamten System zur Folge haben. Der Erfolg wurde bei jeder Durchführung festgehalten. Die ausgefüllten Checklisten sind im Anhang unter Testprotokolle ersichtlich.

4.3. Ergebnisse

Die im Kapitel Anforderungsspezifikation als obligatorisch definierten Use Cases konnten wie geplant im Meilenstein MVP umgesetzt werden. Jedoch verläuft der Bestellprozess nicht wie ursprünglich geplant über eine SAP Instanz. Diese hätte parallel zu dieser Studienarbeit entwickelt werden sollen. Es wurden aber bis zum Implementationsende keine Schnittstellen seitens SAP Instanz zu Verfügung gestellt. Aus diesem Grund wurde eine alternative Lösung entwickelt, welche die Aufgaben des SAP übernimmt und direkt mit der Maschinensteuerung kommuniziert.

4.3.1. Zusätzlich implementierte Features

Direkte Anbindung an Maschinensteuerung
Um die SAP Instanz umgehen zu können, wurde ein zusätzlicher Container (Guerilla) entwickelt, um einen direkten Austausch mit der Maschinensteuerung der Smart Factory zu ermöglichen. Zusätzlich persistiert der Proxy die Bestellungen und bietet Schnittstellen an, um auf diese zugreifen zu können.

Simulierter Bestellprozess
Für Entwicklungs-, Test und Demonstrationszwecke wurde auf dem Order System Proxy ein Mock-Service entwickelt, der den Ablauf einer Bestellung simuliert. Der Unihockeyball wird jedoch nicht produziert.

3D Unihockeyball
Anhand einer bereits existierenden CAD-Zeichnungen eines Unihockeyballes wurde eine animierte und interaktive 3D-Version des Ballles in three.js[15] kreiert. Diese wurde anschliessend im Konfigurator der Bestell-App eingebunden und für dessen Anforderungen angepasst. Zusätzlich wurde ein Check implementiert, der prüft, ob die benötigte Technologie unterstützt wird. Ansonsten wird eine entsprechende Meldung dargestellt.

Filterung der Bestellungen
Um den Nutzer bei der Suche seines Balles zu unterstützen, wurden diverse Filtermöglichkeiten implementiert. Über eine Farbauswahl kann beispielsweise nach einer gewünschten Farbe gefiltert werden. Weiter kann der Benutzer seine Bestellungen auch nach deren Status (Abgeschlossen / in Produktion / bereit zur Abholung) filtern. Zusätzlich wurde ein Suchfeld implementiert, in welchem sowohl Seriennummern von Bällen, als auch von Ballhälften eingegeben werden können. Bei der Eingabe wird sofort mittels Pattern-Matching nach passenden Bällen gesucht. Die gesamte Filterung kann auch mit einem Klick ausgeblendet werden.

Anzeige vom Bestellstatus
Jede Bestellung in der Übersicht ist mit dessen aktuellem Status markiert. Durch ein Polling wird dieser auch ohne neues Laden in der Bestell-App aktualisiert. Zudem wird nur beim Status *bereit zu Abholung* der QR-Code für die Abholung in den Details der Bestellung dargestellt.

4.3.2. Reverse Proxy

Ein Reverse Proxy verarbeitet externe Anfragen und leitet diese an ihre interne Zielressource weiter. Filter und Regeln schränken die Auswahl an möglichen Zielen ein. Als mögliche Eintrittspunkte gibt es drei offene Ports.

Port	Zweck
80 (HTTP)	Leitet eindreffende Nachrichten direkt an den Port *443 (HTTPS)* weiter.
443 (HTTPS)	Anfragen können bei zutreffendem Origin vom Frontend- oder Backend-Container verarbeitet werden.
40008 (HTTP)	Mit Berücksichtigung von Sicherheitsaspekten hätte sich der Konfigurations- und Programmieraufwand seitens Maschinensteuerung für eine funktionierende Schnittstelle gegenüber dem Guerilla stark erhöht. Deshalb wurde gänzlich darauf verzichtet. Da die Maschinensteuerung zudem nicht mit der automatischen HTTPS-Weiterleitung umgehen kann (*Moved Permanently*) wurde für die Guerilla-Schnittstelle zusätzlich ein Bypass für HTTP-Only eingerichtet.

Tabelle 63: Reverse Proxy Eintrittspunkte

Das für HTTPS benötigte Zertifikat wird mittels TLS Challenge[16] von Let's Encrypt und dem ACME-Protokoll[17] automatisch ausgestellt. Dazu wird lediglich ein zur Zieldomain äquivalenter A Eintrag auf dem DNS-Server mit Referenz zum entsprechenden Server benötigt.

4.3.3. Frontend

Die Struktur des Frontends wurde in Seiten und Komponenten aufgeteilt, um eine möglicht hohe Wiederverwendbarkeit der Komponenten zu ermöglichen. Aus diesem Grund wurden auch Styles direkt bei der Komponente abgelegt. Daraus folgte folgende React-Komponenten Struktur.

Pages

Komponente	Beschreibung
Home	Die *Home* Komponente wird als Startseite der Bestell-App verwendet. Sie bietet einen Überblick über deren Aufbau und kann zur Navigation verwendet werden.
Configurator	Hier befindet sich der Konfigurator, beziehungsweise die *Colorpicker* Komponente. Zusätzlich wird die *OrderNavigator* und *OrderConfirmation* Komponente verwendet.
OrderConfirmation	Diese Page ist über keine URL erreichbar, sondern wird lediglich nach einer erfolgreichen Bestellung dargestellt.
Orders	Diese Komponente generiert *Order* Komponenten anhand der getätigten Bestellungen. Zusätzlich wurden darin die Filtermöglichkeiten implementiert.
ProductionData	Beim ersten Laden dieser Page wird nur ein Formular für die Eingabe einer Seriennummer dargestellt. Wird ein Ball anhand dieser Seriennnummer gefunden, wird die *FloorballData* Komponente dargestellt.
Login	Beinhaltet das Login Formular.
Register	Beinhaltet das Registerungs Formular.
About	Beinhaltet ein Video über die Smart Factory.
PageNotFound	Wird eine ungültige URL aufgerufen, wird die *PageNotFound* Page mit einer Fehlermeldung angezeigt.

Tabelle 64: React Pages

Komponenten

Komponente	Beschreibung
AdminSettings	Bietet einem Administrator die Möglichkeit, Bestellungungen auf der Bestell-App zu deaktivieren. Allen Nutzern wird zusätzlich der Zustand der Produktionszelle dargestellt.
BackButton	Stellt einen Button dar, mit dem auf die letztbesuchte Page navigiert werden kann.
ColorPicker	Beinhaltet die gesamte Logik des Konfigurators und besteht aus den Komponenten *Floorball*, *Floorball3D*, *ColorPalette* und *ColorPickerModal*.
Color	Repräsentiert eine einzelne Farbe in der *ColorPalette*.
ColorPalette	Generiert *Color* Komponenten anhand der existierenden Farben und aktiviert sie basierend auf deren Bestand in der Produktionszelle.
ColorPickerModal	Ein Modal das sowohl in der mobilen Ansicht des Konfigurators für die Farbauswahl, als auch für die Farbfilterung in der Bestellübersicht verwendet wird.
ConfirmationModal	Wird verwendet um eine Bestätigung vom Nutzer einzuholen, sobald er eine Bestellung tätigen möchte.
ErrorModal	Wird verwendet, um den Nutzer auf unerwartete Fehler aufmerksam zu machen.
Floorball	2D Version eines Unihockeyballes, der entweder aus dem Wireframe oder den gewünschten Farben besteht.

Floorball3D	Mit three.js implementierte 3D Version der *Floorball* Komponente. Vor der Darstellung prüft sie, ob die technischen Voraussetzungen gegeben sind.
FloorballData	Stellt die Produktionsdaten eines Unihockeyballes und seiner Ballhälften dar.
Footer	Beinhaltet die *AdminSettings* Komponente und einen Link zur *About* Page.
Header	Beinhaltet das OST Logo mit dem Link zur *Home* Page und den Authentifizierungsmöglichkeiten, oder den eingeloggten Nutzer.
OrderNavigator	Bestell Button, welcher den Nutzer durch den Bestellprozess navigiert.
OrderOverview	Beinhaltet Statusinformationen zu einer Bestellung und den QR-Code für die Abholung, sobald der Ball produziert wurde.

Tabelle 65: React Komponenten

Redux
Um die Anzahl Properties der React Komponenten zu reduzieren, wurde Redux als zentrales State Management Tool eingesetzt. Besonders bei verschachtelten Komponenten, wo die Properties ansonsten mehrmals von Parent zu Child Komponenten übergeben werden müssen. Dadurch reduziert sich die Anzahl der benötigten Properties. Dieser zentrale State wird von den Reducern aufgebaut, welche jeweis einen Default State definieren. Verändert wird der Zustandsbaum von Actions, welche die Reducer mittels Dispatcher anstossen und einen neuen Zustandsbaum mit den gewünschten Änderungen aufbauen. Um auf Teile des Zustandbaumes zugreifen zu können, wurden in der Reducer-Datei Getter Funktionen implementiert. Somit müssen Komponenten, die Redux verwenden, den Aufbau des Zustandbaumes nicht kennen. Dank der Hilfsfunktion *MapStateToProps* können die Getter Funktion verwendet werden, um Zustände auf Properties einer gewünschten Komponente zu binden. Zusätzlich können auch gewünschte Actions durch *MapDispatchToProps* als Properties verwendet werden.

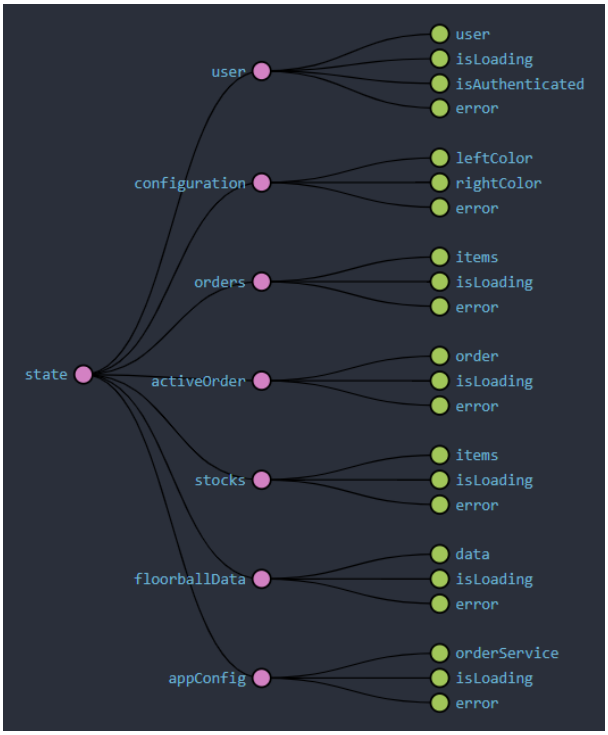

Abbildung 38: Redux Zustandsbaum

Utils
Im Utils-Verzeichnis sammeln sich alle Hilfsfunktionen, welche von Komponenten verwendet werden. Darunter befinden sich Validierungen, Konvertierungen, Formatierungen, Vergleichs- und Hilfsfunktionen für die Umgebungsvariablen. Zusätzlich befindet sich dort die Axios-Instanz, welche als HTTP-Client verwendet wird, um API Aufrufe zu tätigen. Diese enthält zwei Interceptors (Abfänger), welche für die Handhabung von Access- und Refresh Token der Authentisierung zuständig sind. Bei Requests auf die API wird zuerst geprüft, ob der Access Token gültig ist. Ansonsten wird über den Refresh Token ein Neuer angefordert, bevor der Request auf die API ausgeführt wird. Fehlerhafte Antworten der API werden in der Axios-Instanz auf ihren Statuscode geprüft. Bei einem 401 Status (Unauthorized) wird ebenfalls ein neuer Access Token angefordert.

Ressourcen
Alle internen Ressourcen der Bestell-App befinden sich in einem designierten Verzeichnis. Dazu gehören definierte Typen, Styles und Konstanten. Zusätzlich wurden hier alle Strings abgelegt, um die Bestell-App für zusätzliche Sprachen vorzubereiten.

Screenshots
Das Ergebnis des Frontends lässt sich am einfachsten mit Screenshots festhalten. Zu diesem Zweck wurden sowohl die Bildschirme in der Desktop, als auch der mobilen Ansicht festgehalten. Die Responsive Screenshots in der mobilen Ansicht befinden sich im Anhang.

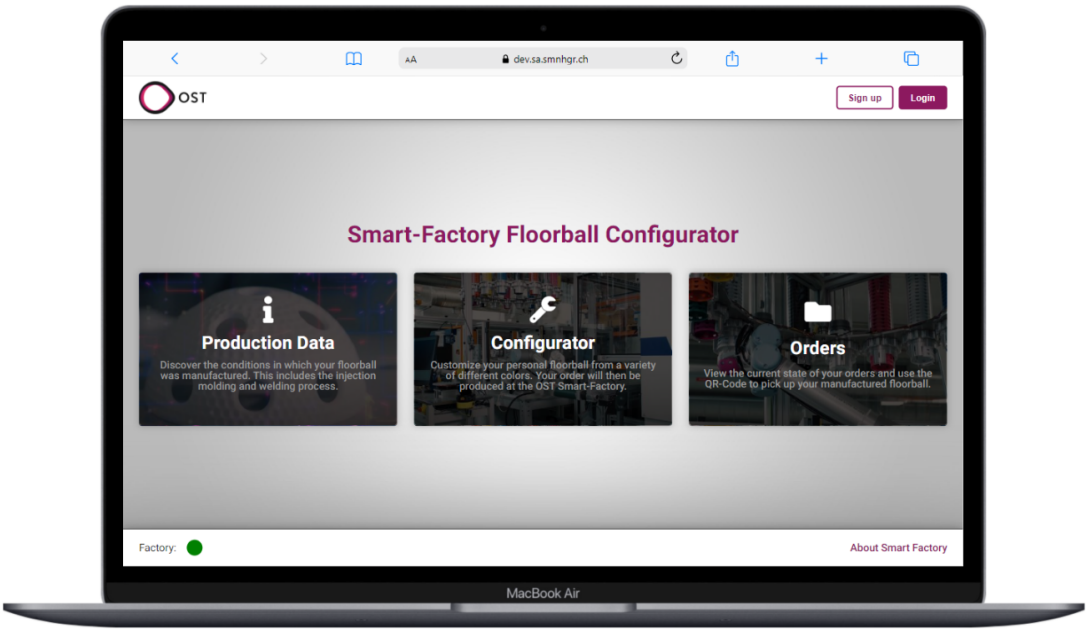

Abbildung 39: Home

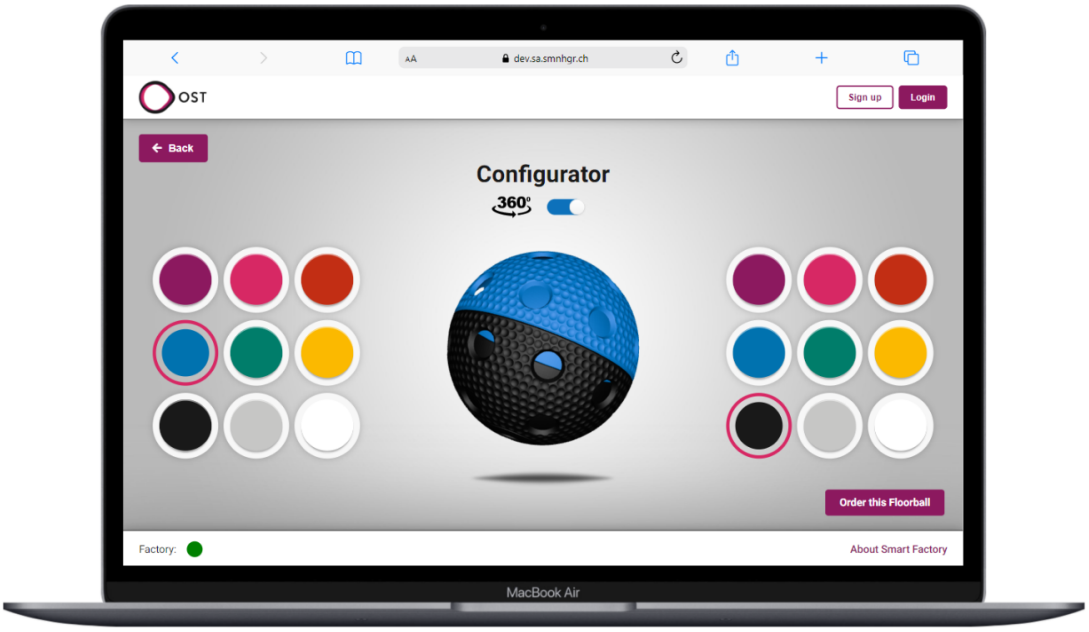

Abbildung 40: Konfigurator

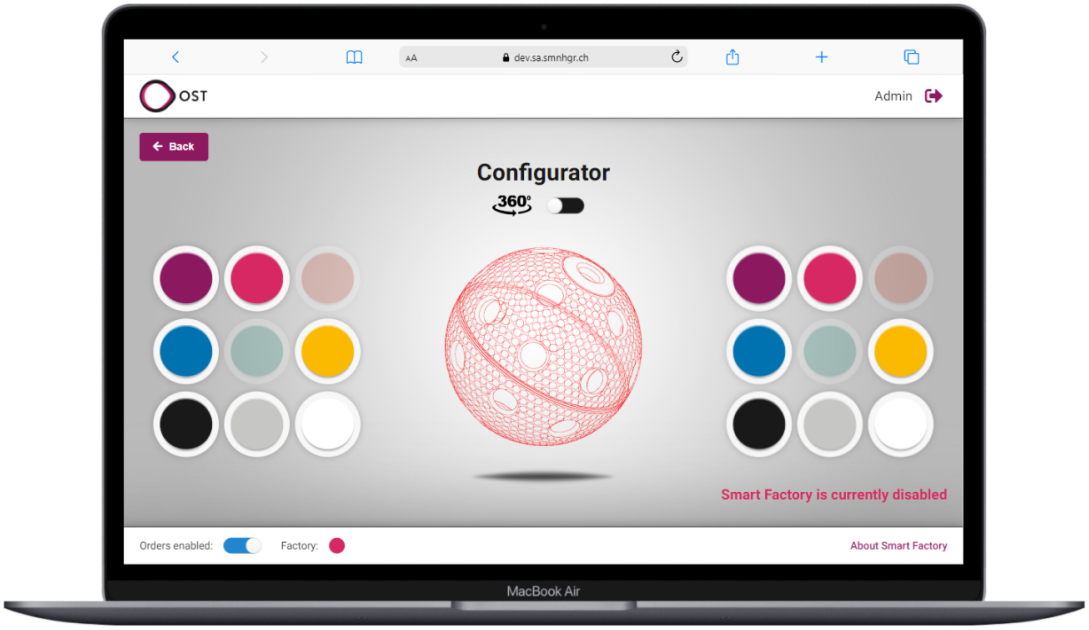

Abbildung 41: Konfigurator deaktiviert

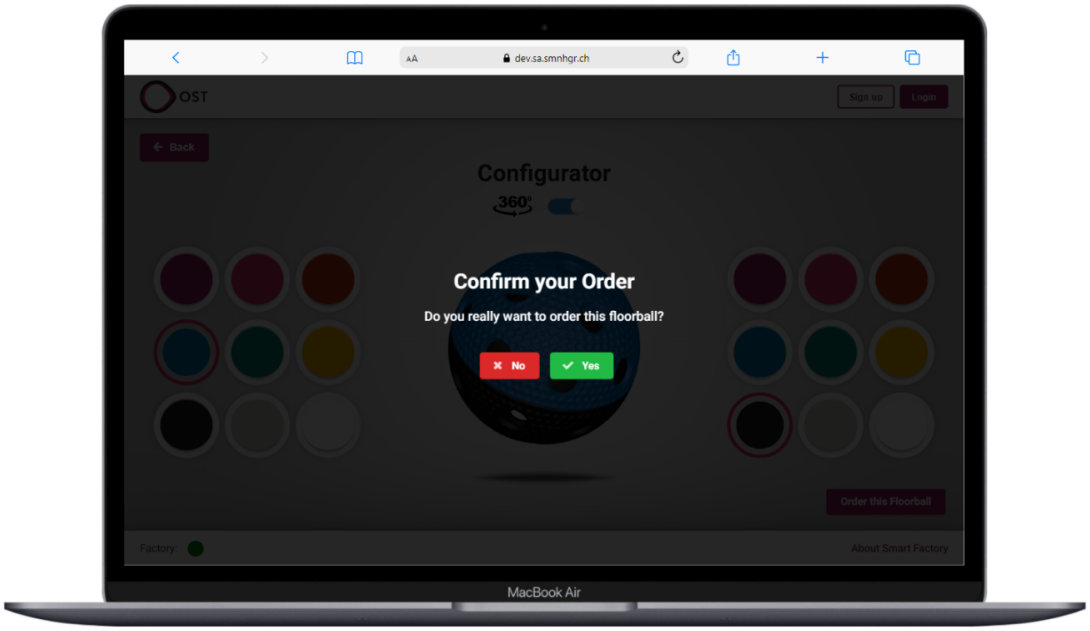

Abbildung 42: Bestätigungs-Modal

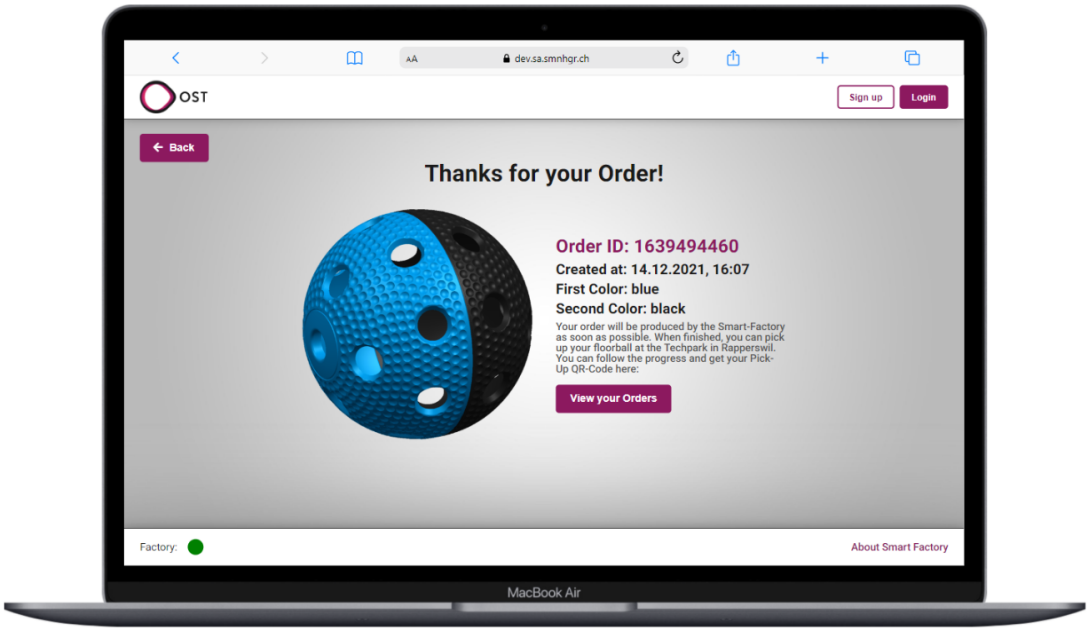

Abbildung 43: Bestellbestätigung

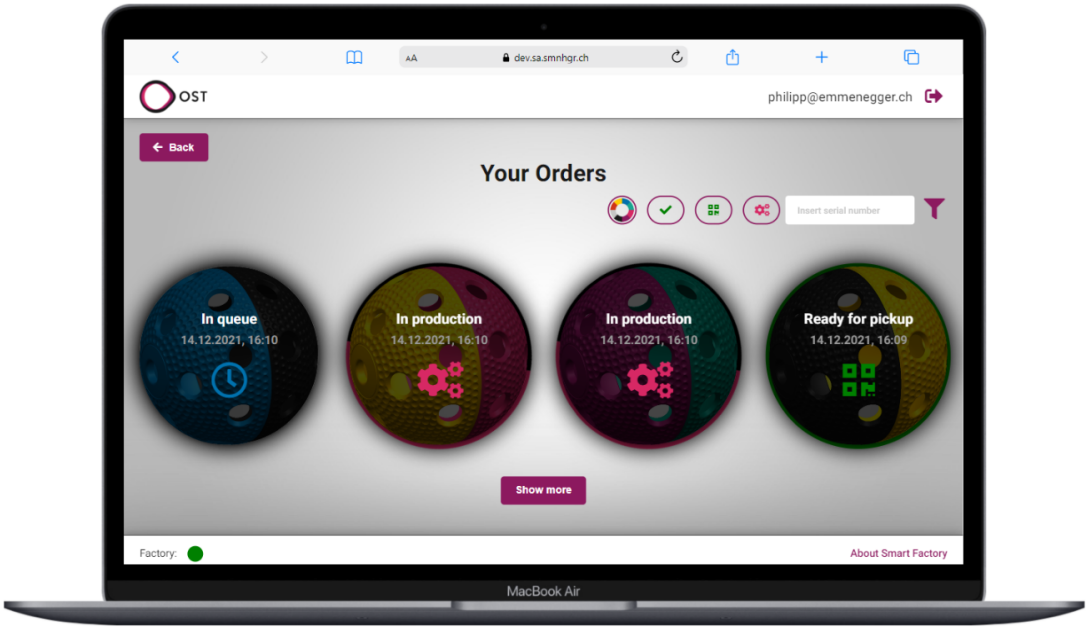

Abbildung 44: Bestellungen

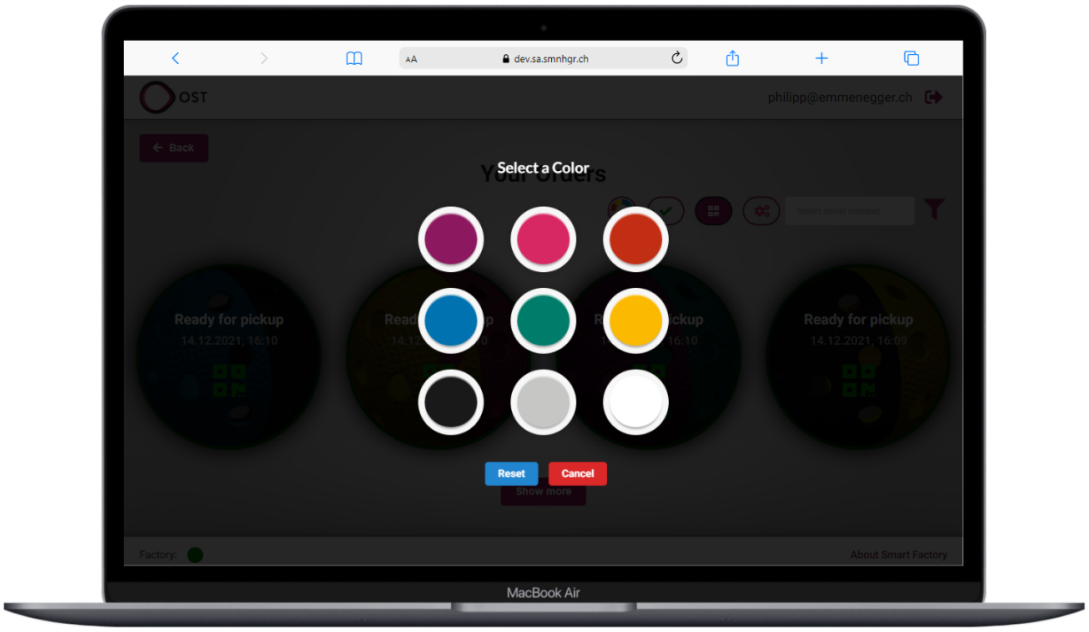

Abbildung 45: Farbfilter

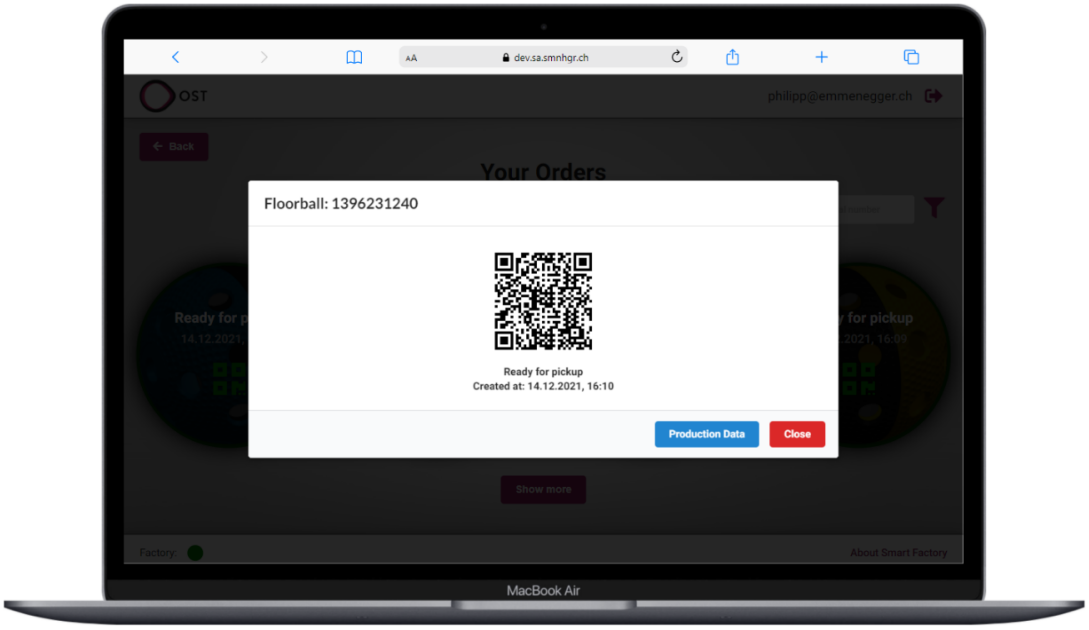

Abbildung 46: Bestellung Details

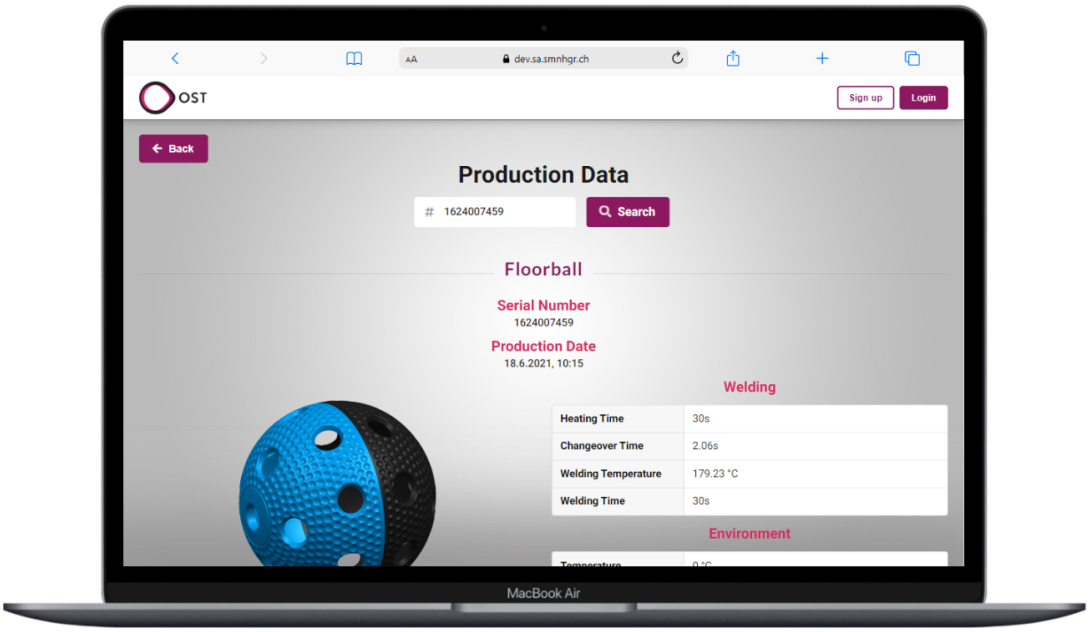

Abbildung 47: Produktionsdaten

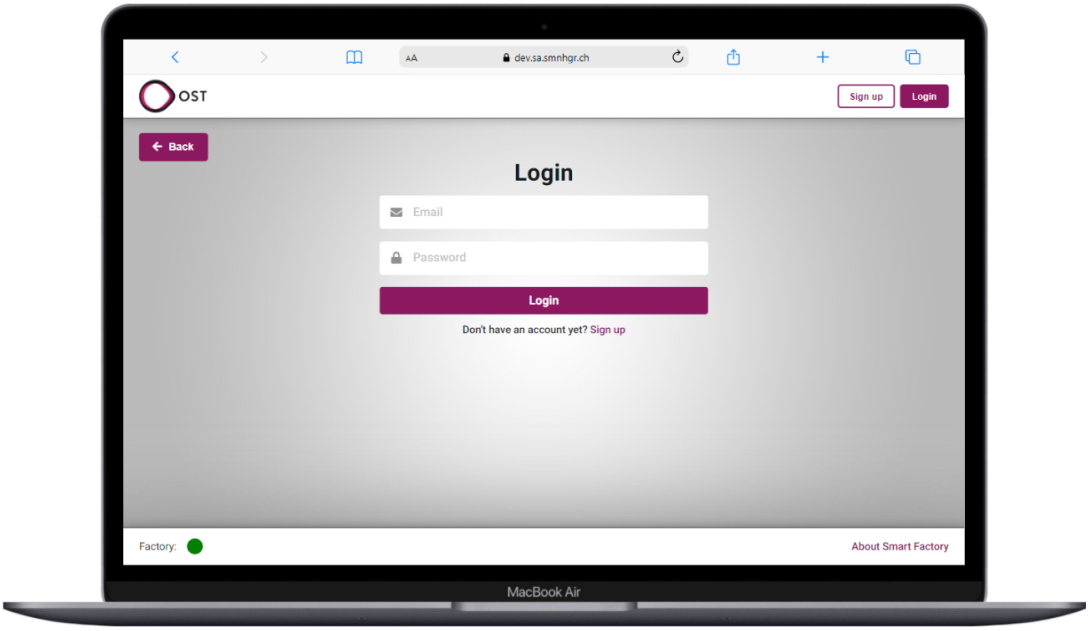

Abbildung 48: Login

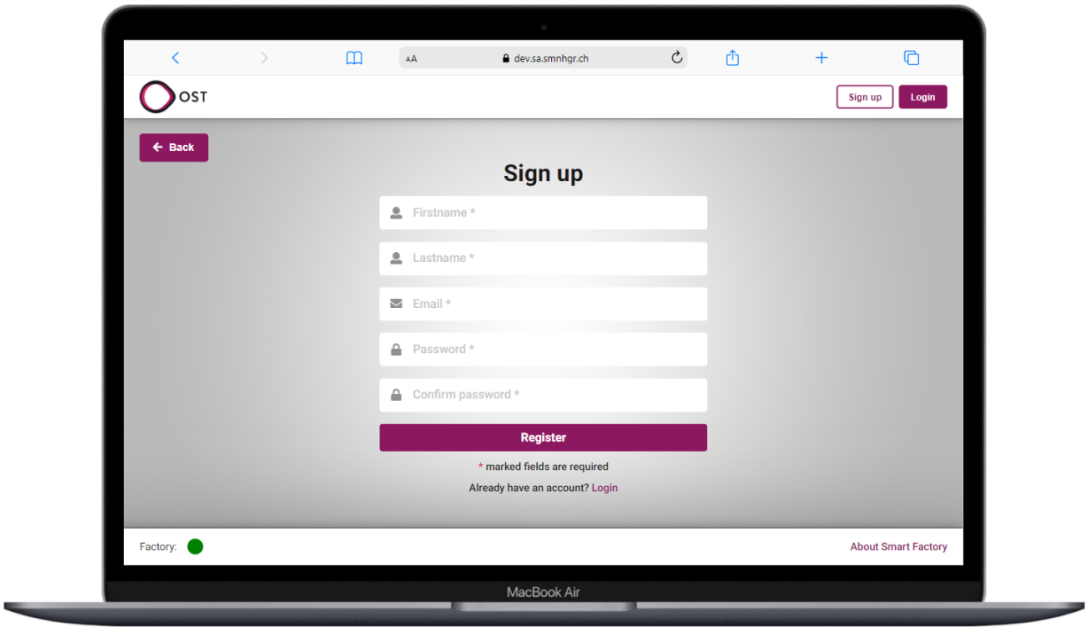

Abbildung 49: Registrieren

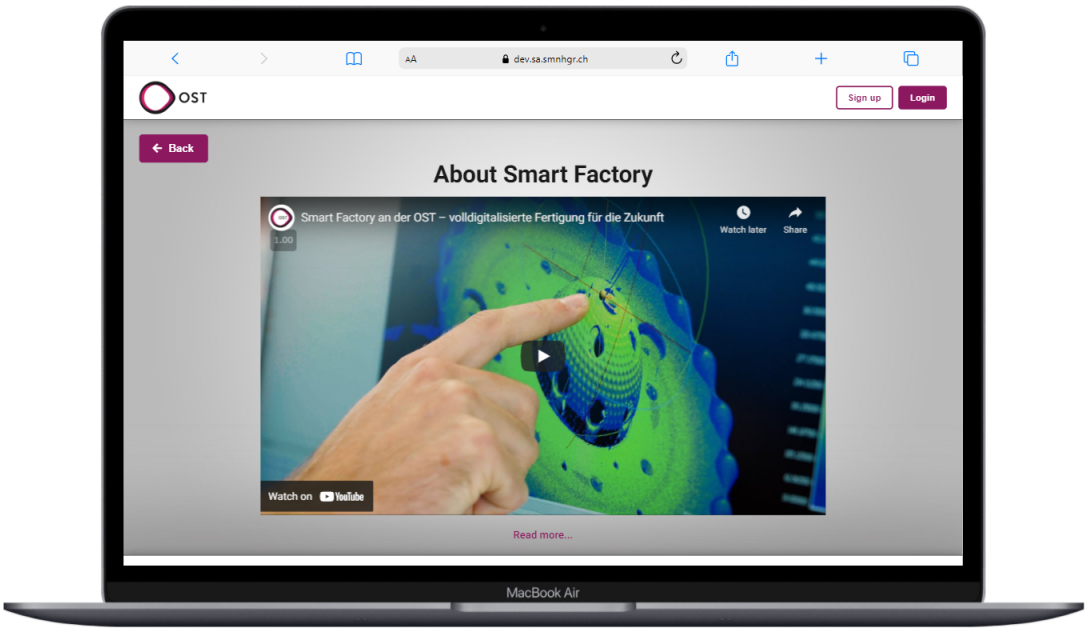

Abbildung 50: About Smart Factory

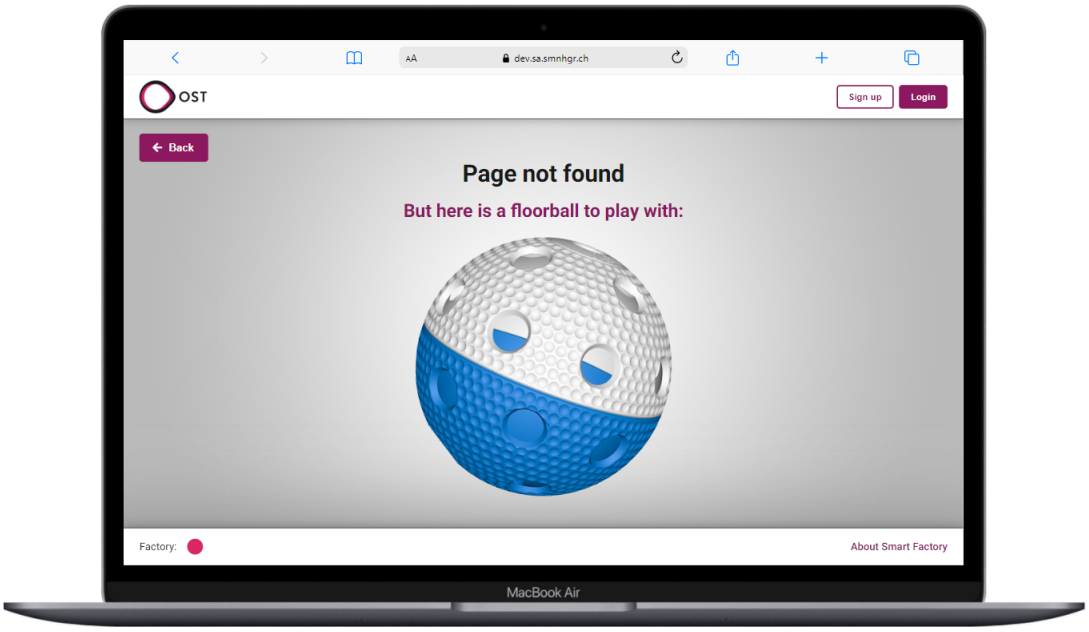

Abbildung 51: Fehlerseite

4.3.4. Api

Die Api ist das Herzstück der Bestell-App. Sie verbindet das Frontend mit dem Bestellsystem und verarbeitet die Authentifizierungsprozesse. Strukturell lässt sie sich grob in Middlewares, Controller, Services und HTTP-Clients unterteilen. Middlewares umschliessen Controller, Controller nutzen Services und Services nutzen wiederum HTTP Clients oder einen DbContext für den Datenzugriff.

Configuration
Über allem stehen die globalen Einstellungen und Konstanten. Volatile Werte können mit Environment Variablen zur Laufzeit gesetzt werden. Alle möglichen Variablen sind in der Datei example.env gelistet.

Variable	Inhalt
ACCESS_TOKEN_... REFRESH_TOKEN_...	Zur Verifizierung des Tokens werden Secrets benötigt. Zusätzlich muss eine Gültigkeitsdauer gewählt werden.
ANONYMOUS_CUSTOMER_...	Der Kunde ohne eigenes Benutzerprofil braucht auch eine Debitorennummer vom Bestellsystem und eine Mailadresse. Diese können damit gesetzt werden.
DB_...	Spezifiziert die Datenbank. Für *inmemory* muss nur der Provider gleichnamig gesetzt werden.
ADMIN_...	Der Benutzername und das Passwort des Administrators können dynamisch angepasst werden. Ohne diese Angaben wird ein Standardbenutzername und -passwort gesetzt.
ORDERSYSTEMPROXY_...	Der Benutzername und das Passwort des Order System Proxy-Benutzers können ebenfalls angepasst werden. Ohne diese Angaben wird ein Standardbenutzername und -passwort gesetzt.

Tabelle 66: Api Environment Variablen

Mit der Authentifizierung werden persönliche Bestellungen ermöglicht. Es wurde eine Authentifizierung mit JSON Web Tokens (JWT) implementiert, womit integere Claims ausgetauscht werden können.

Schema	Inhalt
LoginCredentials	Benutzer können sich mittels *LoginCredentials* gegenüber der Api authentifizieren. Resultat aus dem Login sind ein gültiger Access sowie Refresh Token.
Access Token	Mit einem Access Token kann ein Benutzer zeitlich begrenzt auf geschützte Ressourcen zugreifen. Dazu wird ein gültiger Access Token als Bearer im HTTP-Request-Header gesetzt.
Refresh Token	Läuft der Access Token ab, kann mit einem gültigen Refresh Token ein neuer Access Token verlangt werden.
Basic Authentication	Für interne Systeme steht zusätzlich *Basic Authentication* zur Verfügung.

Tabelle 67: Api Authentifizierungsschemen

Benutzer haben Rollen. Mit bestimmten Rollen können Benutzer zusätzliche Dienste nutzen. Es gibt aktuell drei Rollen.

Rolle	Bedeutung
User	*User* können persönliche Bestellungen aufgeben und verwalten. Ihnen stehen die normalen Funktionen der Bestallapp zur Verfügung.
Admin	Ein Administrator kann Konfigurationen anpassen. Aktuell kann dieser die Bestellmöglichkeit deaktivieren.
Service	Services können ebenfalls die Konfiguration anpassen. Diese Rolle wird ausschliesslich von internen Systemen genutzt.

Tabelle 68: Api Benutzerrollen

Swagger dokumentiert die Schnittstellen der Api. Diese Dokumentation steht zur Laufzeit der Api in sämtlichen Environments unter dem Pfad /swagger/ zur Verfügung.

Die Api kann mit zwei unterschiedlichen Datenbankprovidern gestartet werden. Für das produktive Environment ist zwingend *postgres* zu verwenden.

Provider	Nutzung
postgres	Speichert die Daten in einer relationalen PostgreSQL-Datenbank ab. Dafür ist zwingend eine entsprechende Datenbank zu starten und weitere Parameter richtig zu setzen. Die hinterlegten Datenbankmigrationen stehen ausschliesslich für diesen Datenbankprovider bereit. *postgres* kann sowohl im Environment *Development* als auch in *Production* eingesetzt werden.
inmemory	Speichert Daten nur temporär während einem Lebenszyklus der Api. *inmemory* steht entsprechend nur im Environment *Development* und fürs Testing zur Verfügung.

Tabelle 69: Api Datenbankprovider

Middlewares
Middlewares werden vor einem Controllerauftruf und nach Beendigung eines Controlleraufrufs ausgeführt. Sie erlauben Einfluss auf den HTTP-Request und auf die HTTP-Response.

Middleware	Inhalt
Fehlerbehandlung	In der Api wird bei abweichendem aber erwartbarem Systemverhalten eine *HttpResponseException* geworfen. Mit einer Middleware werden solche Fehler global aufgefangen, behandelt und aufbereitet an den Anfragenden retourniert. Das reduziert die Komplexität einzelner Controller.
Basic Authentication	Damit extern offene Schnittstellen nicht missbraucht werden können, müssen sich auch andere interne Systeme gegenüber der Api mit einem Service-Account authentifizieren. Dafür wurde mittels einer Middleware zusätzlich die Möglichkeit zur *Basic Authentication* geschaffen. Somit muss das Client-System bei entsprechender Anfrage nur einen Benutzernamen und ein Passwort im entsprechenden Header an die Api übermitteln. Der aufwändigere Loginprozess mit Token entfällt.

Tabelle 70: Api Middlewares

Controller
Controller definieren serverseitige Schnittstellen für eingehende HTTP-Request und verarbeiten, beziehungsweise formatieren dessen Eingaben. Sie validieren die Daten serverseitig, authentifizieren den Benutzer und antworten mit geeignetem Inhalt.

Controller	Inhalt
AuthController	Stellt Schnittstellen für den gesamten Lebenszyklus eines Benutzers bereit - von der Registrierung, über das Login bis zum Logout.

ConfigurationController	Darüber lassen sich Einstellungen abfragen und anpassen. Konfigurationsanpassungen werden sowohl vom Frontend, insbesondere dem Administrator als auch von internen Systemen vorgenommen.
HomeController	Verarbeitet die Root-Route und stellt eine zusätzliche Monitoring-/Debugging-Route bereit, welche keine Services benötigt.
OrderController	Damit können Bestellungen aufgegeben und abgerufen werden.
ProductController	Der *ProductController* ruft die Daten eines Unihockeyballes ab.
StockController	Ruft die aktuellen Bestände jedes Typs von Ballhälften ab.

Tabelle 71: Api Controller

HTTP-Clients
Diese übernehmen die HTTP Anfragen an andere Systeme, setzen die passende Route und verarbeiten dessen Antworten. Bestandteil der Api sind zwei HTTP-Clients. Einer für den Order System Proxy und einer für den Production Data Proxy.

Datenbank
Die Api unterhält eine eigene Datenbank. Darin sind zwei Datenbäume abgelegt. Die Benutzerverwaltung und die Konfigurationen.

Tabelle	**Inhalt**
AspNetUsers	Beinhaltet Felder für das Objekt *UserDetails* und von ASP.NET verwendete Felder zur In- und Authentifizierung eines Benutzers.
AspNetUserRoles	Bindet einen Benutzer mit konkreten Rollen.
AspNetRoles	Beinhaltet Felder zur Definition aller Benutzerrollen. Die verfügbaren Rollen werden durch einen initialen Seed gesetzt.
RefreshToken	Beinhaltet Felder zur Definition eines Refresh Tokens. Erhält der Benutzer einen Refresh Token, wird dieser in dieser Tabelle eingetragen.
Configuration	Beinhaltet Felder zur Definition bestimmter Einstellungen. Die initialen Konfigurationseinstellungen werden als Teil der Datenbank-Migration hinzugefügt.

Tabelle 72: Api Datenbanktabellen

Damit bestehende Daten durch Änderungen am Datenbankschema während der Weiterentwicklung nicht verloren gehen und einfach in einen neuen Api-Release überführt werden können, werden Migrationen eingesetzt. Neue, auf das veränderte Schema passende, Migrationen müssen manuell erstellt werden.

4.3.5. Order System Proxy

Die Hauptfunktionalität des Order System Proxy ist die Datenaufbereitung und Datenweiterleitung. Die API tätigt Requests auf den Proxy, welcher die Daten an ein externes System weiterleitet. Die dort generierte Antwort wird wiederum in gewünschter Form an die API retourniert.

Modes
Der Order System Proxy kann in drei unterschiedlichen Modes verwendet werden: *mock, guerilla* und *sap*. Welcher Mode aktiv ist, wird mit der Environment Variable ORDERSYSTEMPROXY_MODE bestimmt. Im Hintergrund werden dann je nach Variable unterschiedliche Services und Datenbankverbindungen gestartet. Hierbei ist zu beachten, dass unter Verwendung von *mock* eine inmemory und unter

guerilla eine PostgreSQL-Datenbank verwendet wird. Entsprechend muss auch die Environment Variable DB_PROVIDER mit *inmemory* oder *postgres* gesetzt sein. Grund für diesen Aufbau war die grosse Unsicherheit, ob die SAP Schnittstellen noch im Laufe des Projekts fertiggestellt werden.

Mode	Zweck
Mock	Im Mock Zustand wird die komplette Ballproduktion simuliert und die Daten werden in einer eigenen Datenbank abgelegt. Die Produktion beinhaltet 4 Status: *inQueue, inProduction, readyForPickup* und *closed*. Wenn eine Bestellung aufgegeben wird, befindet sich der Ball 15 Sekunden in der Queue, daraufhin 30 Sekunden in Produktion und ist dann bereit zur Abholung. Der Status *closed* kann nicht erreicht werden, weil der Ball nicht existiert und entsprechend auch nicht abgeholt werden kann. Beim Start des Proxy werden jedoch zwei Mock-Objekte mit diesem Status angelegt. Dieser Zustand eignet sich besonders gut für Testzwecke und Demonstrationen, bei welchen kein Ball produziert werden soll.
Guerilla	Der Mode *guerilla* wurde entwickelt, damit auch ohne SAP Schnittstellen ein Unihockeyball bestellt werden kann. Der Order System Proxy verwendet dafür zusammen mit dem Guerilla Container eine PostgreSQL-Datenbank. Der Proxy legt dort neue Bestellungen ab und kann Updates vom Guerilla Container auslesen.
SAP	Die Services und der HTTP-Client für das SAP werfen eine NotImplementedException. Sie konnten im Rahmen dieser Arbeit nicht implementiert werden, da die benötigten Schnittstellen dafür nicht rechtzeitig bereitgestellt wurden.

Tabelle 73: Api Datenbanktabellen

4.3.6. Guerilla

Der Guerilla Container bietet den WebService für das SPS an. Zusammen mit dem Order System Proxy teilt sich der Guerilla eine PostgreSQL-Datenbank. Wenn ein Request für offene Bestellungen entgegengenommen wird, durchsucht der Proxy die Datenbank und antwortet mit einer Liste der unbearbeiteten Bestellungen. Die dabei mitgeschickte Floorball ID kann vom SPS wiederum verwendet werden, um Status-Updates der Produktion mitteilen zu können. Der Guerilla Container prüft zudem, ob regelmässige Requests auf die Route */Orders* gemacht werden. So ist es möglich die API zu informieren, ob das SPS aktiv oder inaktiv ist.

Entgegen allen anderen Kommunikationen läuft die Kommunikation zwischen Guerilla Container und SPS über das HTTP Protokoll. Die Kapazität der Mitarbeiter im Techpark reichte nicht aus, um eine verschlüsselte Kommunikation über HTTPS zu implementieren. Da der Ansatz, eine temporäre Notlösung ist, wurde diese Einschränkung so akzeptiert. Zudem kann die SPS Applikation ihre Requests nur als Content Type Text/Plain senden. Der Guerilla Container kann solch einem String verarbeiten, die Bedingung ist aber, dass er exakt dem erwartetem JSON Format entspricht.

4.3.7. Production Data Proxy

Der Production Data Proxy ist für die Kommunikation mit dem Siemens MindSphere zuständig. Seine Aufgabe ist es, Produktionsdaten eines Unihockeyballes vom bestehendem MindSphere-Server auszulesen und sie an die Api passend zurückzuschicken. Dies betrifft einerseits Daten bei der Herstellung von Ballhälften, aber auch solche vom Schweissprozess. Beim Schweissen werden von der Maschine die zwei Seriennummern der Ballhälften ausgelesen und auf die Ball ID gemappt. Der Production Data Proxy kann also anhand einer Ball ID sämtliche Daten über den Schweissprozess, als auch über die Produktion der Ballhälften liefern.

4.4. Einschränkungen

4.4.1. MindSphere

Die Schnittstellen gegenüber der MindSphere Cloud wurden erfolgreich implementiert. Somit können die Produktiondaten eines Balles anhand seiner Seriennummer ausgelesen werden. Ausgeschlossen davon sind Unihockeybälle, welche nicht über die Web App bestellt wurden. Bis zum Zeitpunkt der Abgabe dieser Arbeit werden aber von der Smart Factory aufgrund eines technischen Problems keine neuen Daten in die Cloud gespeichert. Somit kann die Bestell-App keine Daten zu neu produzierten Bällen auslesen.

Um das Feature dennoch demonstrieren zu können, werden zurzeit Produktionsdaten ausgelesen, welche bereits seit einiger Zeit auf der MindSphere Cloud liegen.

Sobald das technische Problem der Smart Factory gelöst wird, ist lediglich eine kleine Anpassung auf dem Production Data Proxy nötig, um die korrekten Balldaten auslesen zu können. Im *ProductionController* auf der Route *GetFloorballData* wird die Seriennummer des Balles überschrieben. Zusätzlich werden im MindSphere Client in der Funktion *GetWeldingData* die Seriennummern der Ballhälften ersetzt. Diese drei Zeilen können entfernt werden, sobald aktuelle Daten in die MindSphere Cloud gespeichert werden, was auch jeweils mit einem Kommentar im Code vermerkt ist.

4.4.2. Anbindung an SAP

In der Aufgabenstellung der Studienarbeit wird die Anbindung an die SAP Instanz der OST als Ziel festgelegt. Diese Anforderung hatte einen wesentlichen Einfluss auf die Planungsphase der Arbeit. Die Architektur wurde darauf ausgelegt, ein externes Bestellsystem verwenden zu können. Im Verlauf der Arbeit wurde jedoch immer klarer, dass die Schnittstellen seitens SAP nicht rechtzeitig implementiert werden. Dieser Umstand wurde im Risikomanagement, als gefährlichstes Risiko erfasst. Gemäss dem dort definierten Eintrittsverhalten wurde daher ein zusäzlicher Container (Guerilla) implementiert, welcher die Bestellungsverwaltung sowie eine Anbindung an die Maschinensteuerung der Smart Factory ermöglicht.

Trotzdem wurde eine Anbindung für SAP vorbereitet, falls die Schnittstellen noch rechtzeitig bereitgestellt werden, oder um die Arbeit weiterführen zu können. Im Order System Proxy wurden sowohl ein HTTP-Client, als auch erforderliche Services vorbereitet.

4.4.3. Subdomain

Zu Beginn des Projektes wurde bei dem Informatikdienst die Subdomain *floorball.i.ost.ch* beantragt. Der Antrag wurde kurz darauf bewilligt und es wurden DNS Einträge für den verwendeten Server erstellt. Jedoch war es mit dieser Ausgangslage nicht möglich, ein Zertifikat von *Let's Encrypt* generieren zu lassen. Trotz zahlreicher Versuche und der Involvierung des Informatikdienstes konnte dieses Problem nicht gelöst werden. Sowohl der vorhandene AAAA Eintrag als auch der CNAME in der DNS Konfiguration konnten aber als Ursache ausgeschlossen werden.

4.4.4. Unbekannte Debitorennummer

Ist bei einem Benutzer in der Api-Datenbank aus der Vergangenheit eine Debitorennummer hinterlegt, welche jedoch für das laufende Bestellsystem unbekannt ist, kann der auftretende Fehler beim Absetzen einer neuen Bestellung oder einem Bestelllistenabruf nicht behoben werden. Die Debitorennummer muss manuell aus der Datenbank gelöscht oder die gesamte Datenbank neu angelegt werden. Dieses Phänomen tritt dann auf, wenn sich der Modus (Mock, Guerilla) und auch die Persistenz der Bestellsystem-Datenbank (InMemory, Postgres) ändert. Das genaue Vorgehen für diesen manuellen Prozess ist im README zur Api beschrieben.

5. Weiterentwicklung

Die ausschlaggebenden Anforderungen an die Studienarbeit konnten erfüllt werden. Im Verlauf der Implementationsphase haben sich jedoch einige Umstände verändert und es sind neue Ideen für Features entstanden. Die dadurch entstandene Veränderung der Priorisierung führte dazu, dass neue Features implementiert und einige der optionalen Use Cases vernachlässigt wurden.

Vernachlässigte obtionale Use Cases:

- UC11: Benutzerdaten ändern
- UC12: Sprache ändern
- UC13: Push-Nachricht senden

Die Bestell-App ist also noch ausbaufähig. Wo möglich wurde bereits darauf geachtet, Erweiterungen zu erleichtern. In den folgenden Abschnitten wird beschrieben, wie die noch nicht implementierten Ideen umgesetzt werden können.

5.1. Mit SAP verbinden

Die SAP Schnittstellen konnten im Zeitraum der Studienarbeit nicht zur Verfügung gestellt werden. Es wurden aber bereits Vorbereitungen getroffen, um eine Anbindung rasch implementieren zu können. Sobald die benötigten Schnittstellen zur Verfügung stehen, müssen lediglich die vorbereiteten Services und der Sap-Client im Order System Proxy implementiert werden.

5.2. Ausbauen zu einer PWA

Um die Installierbarkeit der Bestell-App zu ermöglichen, sind einige Schritte nötig. Das Manifest, welches dazu benötigt wird, existiert bereits. Dieses beschreibt die wichtigsten Eigenschaften einer Web App. Folglich muss in einem ersten Schritt ein ServiceWorker registriert werden. Durch ihn ist es möglich, das Verhalten der Applikation in jeder Netzwerksituation zu steuern. Ressourcen können durch ihn im Cache abgelegt werden, um gewisse offline Funktionalitäten anbieten zu können.

5.3. Push-Benachrichtigungen versenden

Sobald die Bestell-App zu einer PWA ausgebaut ist, können Push-Benachrichtigungen bei Statusänderungen den Bestellungen hinzugefügt werden. Dazu muss der ServiceWorker der PWA ergänzt und die Berechtigung vom Nutzer eingeholt werden. Anschliessend können die Benachrichtigungen über einen PushManager registriert werden, damit der Server diese dem Nutzer zustellen kann.

5.4. Mehrsprachigkeit

Die zurzeit ausschliesslich englischen Texte der gesamten Bestell-App wurden bereits in eine Ressourcen Datei ausgelagert. Für die Implementation der Mehrsprachigkeit, muss diese Datei in die gewünschten Sprachen übersetzt werden. Es stehen bereits verschiedene Tools zur Verfügung, um anschliessend anhand der Sprachauswahl des Nutzers die korrekte Ressourcen Datei zu laden.

5.5. Benutzerdaten ändern

Bisher ist es dem Nutzer nicht möglich nach der Registration seine Benutzerdaten anzupassen. Dafür müsste eine zusätzliche Seite im Frontend erstellt werden, wo er diese anpassen kann. Um die Änderungen persistieren zu können, müssen im Backend weitere Routen implementiert werden.

5.6. Sicherheit erhöhen

Grundlegende Sicherheitsaspekte konnten im aktuellen Release bereits berücksichtigt werden. Trotzdem gibt es noch einige Schwachstellen, die es zu beheben gibt. Die definierten Cors Richtlinie erlauben zurzeit

von jedem Client, Anfragen an die Api zu tätigen. Deshalb soll der Origin zukünftig auf die Frontendadresse eingeschränkt werden. Dafür soll die Cors Policy in der Api entfernt und strikter auf dem Reverse Proxy gesetzt werden.

Die Bestell-App und speziell die Api ist momentan teilweise vor Brute Force Angriffen ungeschützt. So wäre beispielsweise ein Missbrauch der Email-Verfügbarkeitsprüfung zur Evaluierung sämtlich registrierter Mailadressen möglich.

6. Projektplan (Soll)

6.1. Einführung

6.1.1. Zweck

Dieses Kapitel beschreibt den Projektplan und bietet einen Überblick über die Studienarbeit Unihockey-Ball Bestell-App. Es liefert erste Informationen zur zeitlichen Planung, der Organisation, zu möglichen Risiken und Qualitätsmanagement. Mit dem Projektplan wird die Basis für eine erfolgreiche Projektabwicklung gelegt.

6.1.2. Gültigkeitsbereich

Der Gültigkeitsbereich erstreckt sich über die ganze Projektdauer innerhalb des Herbstsemesters 2021. Das Dokument wird laufend mit neuen Informationen ergänzt und im Versionsverlauf festgehalten.

6.1.3. Produkt

- Anforderungsspezifikation
- Domainanalyse
- Testspezifikation und Testprotokolle
- Wireframes und Design-Dokumentation
- Softwarearchitektur
- Deployment-Dokumentation

6.2. Projektorganisation

Alle wichtigen Entscheidungen werden gemeinsam im Projektteam getroffen. Aus diesem Grund wird auf eine flache Hierarchie gesetzt, bei welcher alle Mitglieder gleichgestellt sind und sich gleichermassen am Projekt beteiligen. Lukas Kretschmar steht als Betreuer zur Seite und nimmt die Bewertung vor.

6.2.1. Organisationsstruktur

Name	Email	Rolle
Simon Hager	simon.hager@ost.ch	Teammitglied
Philipp Emmenegger	philipp.emmenegger@ost.ch	Teammitglied
Joel Schaltegger	joel.schaltegger@ost.ch	Teammitglied
Lukas Kretschmar	lukas.kretschmar@ost.ch	Betreuer / Product Owner

Tabelle 74: Organisationsstruktur

6.2.2. Externe Schnittstellen

Name	Email	Rolle
Curdin Wick	curdin.wick@ost.ch	Ansprechsperson Produktionszelle Techpark
Bernd Leonhardt	bernd.leonhardt@ost.ch	Ansprechsperson SAP-Schnittstelle
Prof. Dr. Roman Hänggi	roman.haenggi@ost.ch	Ansprechsperson Strategie Smart Factory
Prof. Dr. Stefan Stöckler	stefan.stoeckler@ost.ch	Betreuer SAP-Studienarbeit

Tabelle 75: Externe Schnittstellen

6.3. Zeitmanagement

6.3.1. Zeitraum

Der offizielle Projektstart war am 20. September 2021 und endet am 24. Dezember 2021 um 17 Uhr mit der Abgabe der Studienarbeit.

Projektdauer	14 Wochen
Projektstart	Montag, 20. September 2021
Projektende	Freitag, 24. Dezember 2021 17 Uhr

Tabelle 76: Zeitraum

6.3.2. Zeitbudget

Jedes Teammitglied ist mit jeweils 17.1 Stunden pro Woche oder insgesamt 240 Arbeitsstunden gleichermassen an der erfolgreichen Entwicklung des Produktes beteiligt. Insgesamt stehen dem Projekt somit 720 Stunden zur Verfügung.

Anzahl Projektmitarbeiter	3 Personen
Arbeitsstunden pro Woche und Person	17.1h
Arbeitsstunden insgesamt	720h

Tabelle 77: Zeitbudget

6.3.3. Zeitplanung

Die Zeitplanung und die Arbeitspakete werden direkt im GitLab verwaltet. Sie unterliegt einem agilen Projektmanagement und wird kontinuierlich auf neue Anforderungen und Erkenntnisse angepasst. Veranschlagte und effektive Zeitaufwände jedes Arbeitspaketes werden auf 15 Minuten genau im GitLab erfasst.

In diesem Projekt wird Scrum+ eingesetzt, eine Kombination aus Rational Unified Process (RUP) und Scrum. Mit RUP wird das Projekt zeitlich grob in die Phasen Inception, Elaboration, Construction und Transition unterteilt. Innerhalb dieser Phasen kommt das agilere Projektmanagement Scrum zum Einsatz. Der Projektzeitraum wird in zweiwöchige Sprints unterteilt. In diesem Projekt wird jeder Sprint mit einem Meilenstein und anschliessendem Review abgeschlossen. Das Projektteam macht vor jedem neuen Sprint ein Sprint Planning und führt nach jedem abgeschlossenen Sprint ein internes Sprint Review durch.

Phase	Start	Ende
Inception	20.09.2021 (SW01)	01.10.2021 (SW02)
Elaboration	02.10.2021 (SW02)	22.10.2021 (SW05)
Construction	23.10.2021 (SW05)	17.12.2021 (SW13)
Transition	18.12.2021 (SW13)	24.12.2021 (SW14)

Tabelle 78: Projektphasen

Abbildung 52: Zeitmanagement Übersicht

6.4. Meilensteine

6.4.1. M0: Kick off - 23.09.2021

Das Projekt ist gestartet und die erste Sitzung mit dem Betreuer wurde durchgeführt. In dieser Besprechung wurde die Aufgabenstellung und grob das weitere Vorgehen besprochen.

Arbeitsprodukte (Dokumente) sind:

- Aufgabenstellung

6.4.2. M1: Project plan - 01.10.2021

Der Projektplan gibt dem Projekt die Struktur für einen erfolgreichen und termingerechten Abschluss.

Arbeitsprodukte (Dokumente) sind:

- Projektplan
- Risikoanalyse

6.4.3. M2: Requirements - 08.10.2021

Anforderungen an das Produkt sind spezifiziert und eine Domainanalyse wurde erstellt. Stakeholder, insbesondere der Kunde, wurden in die Spezifizierung einbezogen.

Arbeitsprodukte (Dokumente) sind:

- Anforderungsspezifikation
- Domainanalyse
- Product Backlog (Epics, User Stories)

6.4.4. M3: End of Elaboration - 22.10.2021

Arbeitsumgebung ist vollständig eingerichtet. Ein lauffähiger Prototyp (vertikale Integration) wurde erstellt und geprüft (Proof of Concept). Die meisten technischen Risiken wurden minimiert oder vollständig eliminiert.

Arbeitsprodukte (Dokumente) sind:

- Systemarchitekturentwurf
- Prototyp (Proof of Concept)
- Deploymentstrategie

6.4.5. M4: Architektur - 29.10.2021

Die physische und logische Architektur/das Design ist stabil und vollständig dokumentiert. Designentscheidungen für entwickelte Schnittstellen, Klassenstrukturen sowie eingesetzte Technologien, Persistenz-, UI- und UX-Lösungen sind begründet.

Arbeitsprodukte (Dokumente) sind:

- Systemarchitektur
- Wireframes

6.4.6. M5: MVP (Alpha-Release) - 12.11.2021

Das MVP, ein minimal brauchbares Endprodukt, steht. Eine mit den nötigsten Kernfunktionen ausgestattete erste Version ist online (im Web verfügbar). Qualitätsmassnahmen wurden im erforderlichen Mass befolgt. Systemtests sind durchgeführt und Coderichtlinien eingehalten.

Das MVP beinhaltet die Use Cases

- Benutzer kann einen Ball konfigurieren
- Benutzer kann eine Bestellung platzieren
- Benutzer kann Bestellung abholen (QR-Code abrufen)

Sämtliche Abhängigkeiten zu externen Systemen sind implementiert und getestet. Somit ergeben sich folgende Sekundärziele

- Erfolgreiche Kommunikation mit SAP ist sichergestellt
- Erste Systemtests durchgeführt

Arbeitsprodukte (Dokumente) sind:

- MVP
- Qualitätsnachweis

6.4.7. M6: Feature Freeze (Beta-Release) - 10.12.2021

Das Produkt hat seine finale Grösse innerhalb dieses Projektes erreicht. Es werden keine neuen Funktionalitäten mehr hinzugefügt sondern nur noch die Implementierung der bestehenden Funktionen verbessert und Fehler behoben.

Arbeitsprodukte (Dokumente) sind:

- Beta-Release Product

6.4.8. M7: Final Submission - 24.12.2021 17 Uhr

Das Projekt ist abgeschlossen. Alle während der Arbeit erstellten Dokumente und der gesamte Quellcode werden eingereicht indem diese auf avt.i.ost.ch hochgeladen werden.

Arbeitsprodukte (Dokumente) sind:

- Dokumentation (Schlussbericht)
- Eigenständigkeitserklärung
- Software Repository mit ganzem Code und allen Versionen
- Plakat
- Deployment-Dokumentation

6.5. Iterationen (Sprints)

Sprint	Produkt (Software)	Projekt (Dokumentation)
1	• Projektmanagementtools einrichten • Aufgabenstellung verstehen • Vorarbeiten untersuchen	• Projektplan erstellen • Risikoanalyse durchführen
2	• Product Backlog füllen • Entwicklungsumgebung (CI/CD, Build-/Test-Prozess) einrichten • Schnittstellen testen • Architekturprototyp entwickeln	• Anforderungen spezifizieren • Domain-Modell erstellen • Architekturentwurf erstellen • Technologiestack definieren • UC Siemens Mendix evaluieren
3	• Systemarchitektur bauen • Mocks schreiben • Deployment testen	• Wireframes erstellen • Architektur ausarbeiten • Proof of Concept aufzeigen • Deploymentstrategie entwickeln • Risikoanalyse aktualisieren
4	• Wireframes realisieren • UC Ball konfigurieren implementieren • UC Bestellung platzieren implementieren • UC Bestellung abholen implementieren	• Wireframes validieren

5	• Code verbessern/optimieren • UC Authentifizierung implementieren • UC Ball-Daten abfragen implementieren • UC Bestellübersicht implementieren	
6	• Zusätzliche Features implementieren	• Deployment beschreiben
7	• Code verbessern/optimieren (Refactoring) • Bugs und Fehler beheben • Reserve	• Abstract • Management Summary • Projektmonitoring (Ist) • Fazit/Reflexion • Eigenständigkeitserklärung • Plakat

Tabelle 79: Iterationen

6.6. Besprechungen

6.6.1. Besprechungen mit Betreuer

Alle zwei Wochen, jeweils am Donnerstag trifft sich das Team (wenn möglich am Campus, ansonsten über Teams) mit dem Projekt-Betreuer für ein einstündiges Meeting. Darin wird sowohl der Fortschritt im Projekt, sowie das weitere Vorgehen besprochen.

In den übrigen Wochen wird dem Betreuer im informellen Rahmen kurz der Zwischenstand präsentiert, falls dies einen Mehrwehrt erzielt. Ansonsten werden diese Besprechungen nicht durchgeführt.

6.6.2. Sprint Planning

Bei jedem neuen Sprint werden im Projekt-Team jeweils am Montag Sprint Planning Meetings gehalten. Darin werden Fortschritte reflektiert und das weitere Vorgehen für den kommenden Sprint geplant.

- Review
  - Letzten Sprint abschliessen
  - Über unbeendete Arbeitspakete entscheiden
- Retrospektive
- Sprint-Ziel definieren
- Arbeitspakete planen

6.6.3. Daily Scrum

An Arbeitstagen (Montag, Donnerstag, Freitag und eventuell Wochenende) werden kurze Besprechungen gehalten, um Fortschritte innerhalb vom Projektteam zu kommunizieren und um den Wissensstand zu synchronisieren.

- Jeder Projektmitarbeiter beantwortet die drei Fragen
  - Was habe ich seit dem letzten Meeting gemacht?
  - Was muss ich noch bis zum nächsten Mal erledigen?
  - Gibt es irgendwelche Probleme, Blockaden oder Hindernisse, die mich daran hindern meine Ziele zu erreichen?
- Grössere Probleme diskutieren, kleinere Probleme adressieren

6.7. Risikomanagement

6.7.1. Risiken

Um die Risiken und ihren potentiellen Schaden besser einschätzen zu können, sind diese anhand ihrem maximalen Schadenspotential [h] und ihrer möglichen Eintrittswahrscheinlichkeit aufgelistet und visualisiert. Alle Zahlen sind Schätzungen und sind nicht als verbindlich zu werten.

Nr	Beschreibung	[h]	%	Gewichtung
R1	SAP Schnittstelle nicht oder unvollständig vorhanden	30	60%	18
R2	MindSphere Schnittstellen unklar	15	60%	9
R3	Mangelnde Qualität der Vorarbeiten führen zu Mehraufwand	18	40%	7.2
R4	Unpassende Auswahl der Softwarearchitektur	30	20%	6
R5	Zu hohe Komplexität der Funktionalität	15	30%	4.5
R6	Hohe Komplexität des Loginprozesses	12	40%	4.8
R7	Anforderungen der Coding-Guidelines nicht erfüllt	10	30%	3
R8	Fehlende / fehlerhafte Tests	20	45%	9
R9	Probleme beim Aufsetzen der Softwarekomponenten	15	40%	6
R10	Probleme beim Aufsetzen der CI / CD Pipelines	12	50%	6
R11	Nichteinhalten von Meilensteinen / Deadlines	20	25%	5
R12	Unterbrüche in der Verfügbarkeit der OST Infrastruktur	10	50%	5
R13	Mangelnde Kommunikation im Team führt zu Missverständnissen	30	60%	18
R14	Sicherheitslücken in der Software	20	20%	4
R15	Mangelnde Technologie-Kenntnisse im Team	20	40%	8
				Total: 113.5

Tabelle 80: Risiken

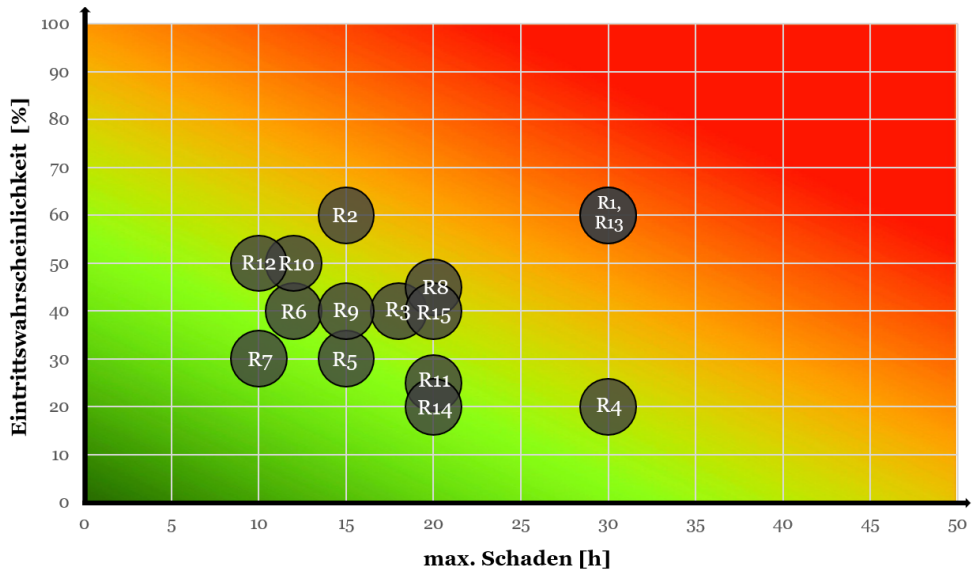

Abbildung 53: Risikoanalyse

6.7.2. Umgang mit Risiken

Nr	Vorbeugung	Verhalten beim Eintreten
R1	Abstraktion der Schnittstelle im Backend	Mocking der SAP Schnittstelle und Produktionszelle direkt ansteuern
R2	Einlesen in Möglichkeiten und API von MindSphere	Siemens Support anfragen
R3	Vorarbeiten studieren, um Mängel früh zu erkennen	Verbesserungen anbringen oder Alternativen ausarbeiten
R4	Umfassende Planung bei der Wahl von Architektur-Bestandteilen	Kompromiss / Alternative für unpassenden Bestandteil finden
R5	Komplexität der Funktionen in der Planung realistisch einschätzen	Funktionalität einschränken, auf MVP konzentrieren
R6	Loginprozess mit SAP-Verantwortlichen im Voraus absprechen	Lösung mit SAP-Team entwickeln
R7	Coding Guidelines festlegen, Code-Reviews durchführen	In Code-Review ansprechen, aus den Fehlern lernen
R8	Umfassende, strukturierte Testpläne schreiben	Testfälle ergänzen, Testcoverage erhöhen, Test-Reviews durchführen
R9	Nur verbreitete Software mit gutem Support verwenden	Komponente ersetzen
R10	Dokumentation von GitLab CI / CD studieren	Unterstützung von Experten anfragen (INS)
R11	Zeitplanung erstellen und laufend aktualisieren, Probleme frühzeitig kommunizieren	Prioritäten setzen, unwichtiges vernachlässigen
R12	Lokale Backups aller Daten	Eigener GitLab Runner entwickeln
R13	Periodische Meetings und Informationskanal (MS Teams) definieren	Zusätzliche Besprechungen festlegen
R14	Aktuelle Versionen von Komponenten verwenden, bekannte Sicherheitslücken meiden	Sicherheitslücke adressieren und beheben
R15	Fähigkeiten trainieren in Testumgebung, Einlesen in neue Technologien	Expertenmeinung einholen

Tabelle 81: Umgang mit Risiken

6.8. Arbeitspakete

Die Arbeitspakte wurden anhand der Produkte in den jeweiligen Meilensteinen grob definiert und als Issues auf GitLab erfasst. Mit Labels werden sie den verschiedenen Sprints (1 - 7) zugeordnet und mit relativen Aufwänden eingeschätzt. Während den Sprint Planning Besprechungen werden die groben Arbeitspakete des nächsten Sprints in kleinere, spezifischere Pakete aufgeteilt und diese mit einer absoluten Aufwandsschätzung versehen.

6.9. Qualitätsmassnahmen

Um eine hohe Qualität zu garantieren, werden folgende Massnahmen getroffen:

Massnahme	Zeitraum	Ziel
Sprint Reviews	Ende Sprint	Austausch über den aktuellen Stand, um Engpässe und Probleme frühzeitig zu erkennen
Meilensteine	Fixer Tag	Sicherstellen, dass das Projekt erfolgreich umgesetzt wird
Code Reviews	Merge Request	Clean Coding sicherstellen
Automatisierte Tests	Push	Funktionalität von Code testen
Testdriven Development	Coding	Ungetesteter Code verhindern
Definition of done	Schliessen von Issue	Für jedes Issue wird folgende DoD eingehalten

Tabelle 82: Qualitätsmassnahmen

6.9.1. Definition of done

- CI erfolgreich durchlaufen
- Coding Guidelines eingehalten
- Code Review erfolgreich
- Unit Test Abdeckung
- Wenn nötig: Dokumentation aktualisiert

6.9.2. Dokumentation

Die Dokumentation ist auf dem GitLab Server der OST abgelegt. Bei jedem Merge in den Master Branch wird das PDF als Artifakt neu kompiliert. So ist die Versionierung sichergestellt.

6.9.3. Projektmanagement

Auch für das Projektmanagement wird GitLab verwendet. Zu jedem Task wird ein Issue erstellt und darin mit */estimate* beziehungsweise */spend* der Aufwand geschätzt und getrackt. Der Repository-Time Report wertet diese Daten jede Nacht um 2 Uhr aus und erstellt drei Grafiken über die Zeitverteilung unter den Teammitgliedern, den Meilensteinen und in welchen Tätigkeiten diese aufgewendet wurden.

6.9.4. Unit Testing

Über GitLab CI/CD wird bei jedem Commit-Push auf den Server die Pipeline durchlaufen. Damit wird vor jeder Änderung geprüft, ob sich der Code builden lässt und ob die Testfälle erfolgreich sind.

6.9.5. Code Reviews

Bei jedem Merge Request wird ein Code Review durchgeführt, bei welchem das zuständige Projektmitglied überprüft, ob alle Qualitätsmassnahmen eingehalten wurden.

6.9.6. Code Style Guides

Für die Einhaltung von unserer Code Style Guides werden Code Formatting Tools eingesetzt. Durch sie wird anhand der definierten Regeln automatisch ein stilistisch einheitlicher Code produziert.

6.9.7. Namenskonvention

Die Issues (Tasks), insbesondere dessen Titel, unterliegen folgender Struktur:

Titel	**Beschreibung**	**Beispiele**
Sprache	Die Titel der Issues werden in englischer Sprache verfasst	Research SAP
Satzbau	Die Titel sind immer nach dem Schema: beschreibendes Verb + Nomen + Übriges aufgebaut. Ausnahme Besprechungen. Es werden keine Modalverben eingesetzt.	Design Wireframes, Evaluate Interface, Develop Service, Test Prototype
Gross- /Kleinschreibung	Die Titel, das Verb, der Issues beginnen immer mit einem Grossbuchstaben	Setup Server
Zuweisung	Da Issues in unserer GitLab-Version nicht an mehrere Teammiglieder zugewiesen werden können, werden Titel von Gruppenaufgaben mit dem Präfix *All:* ergänzt	All: Kickoff Meeting
Abkürzungen	Wörter im Titel werden vollständig ausgeschrieben	
Textlänge	Der Titel ist so kurz wie möglich und so lange wie nötig, um das Issue eindeutig zu identifizieren	Write Section Projectplan

Tabelle 83: Issue Namenskonvention

7. Projektmonitoring (Ist)

7.1. Zeitauswertung

7.1.1. Arbeitsverteilung

Die folgende Grafik zeigt den Aufwand in Stunden pro Projektmitglied.

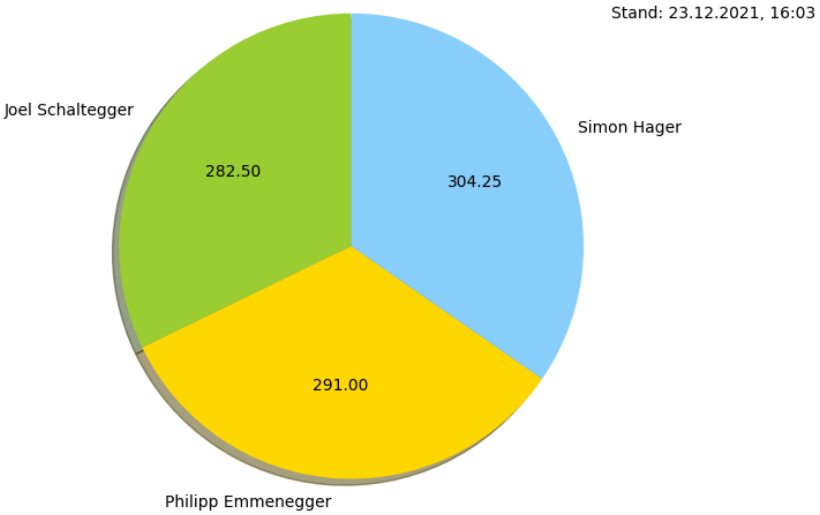

Abbildung 54: Time-Report Mitglied

7.1.2. Arbeitsbereich

Jedes Issue wurde eines der sechs Labels zugewiesen.

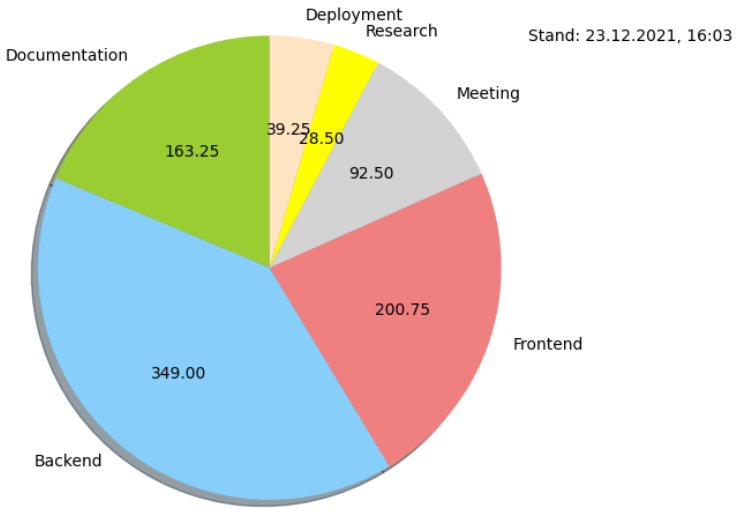

Abbildung 55: Time-Report Labels

7.1.3. Soll-Ist-Zeitvergleich

Jeder Aufwand wurde zeitlich geschätzt und einem Meilenstein zugeordnet. Auf den ersten Blick könnte man die Annahme treffen, dass das Team gegen Ende des Projekts viel Mehraufwand leisten musste. Der Meilenstein 6 - Feature Freeze streckte sich jedoch über einen doppelt so langen Zeitraum wie der vorherige. Das Selbe gilt auch für den Start des Projekts. Dort gab es noch kürzere Zeitintervalle.

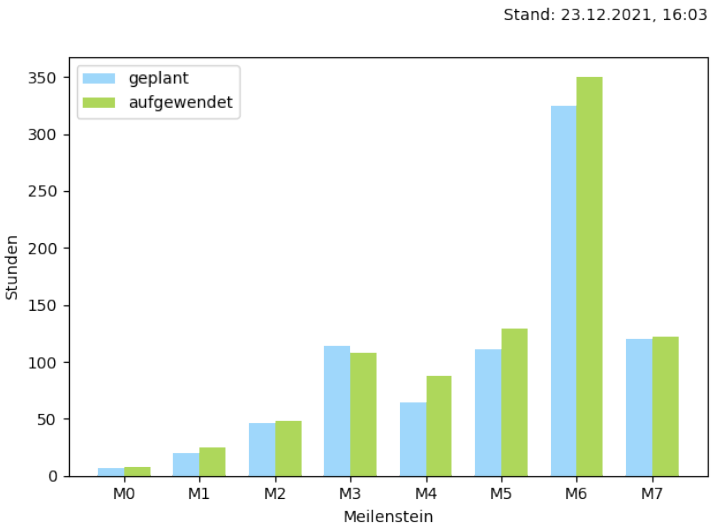

Abbildung 56: Time-Report Meilensteine

7.1.4. Meilenstein-Einhaltung

Im Verlauf der Arbeit konnten alle Meilensteine eingehalten werden. Das beweist, dass die Projektplanung und allgemein die zeitliche Einschätzung zu Beginn des Projekts gut war. Das Team hat sich auch stets gegenseitig unterstützt, falls jemand in Zeitnot geraten ist.

7.2. Implementierungsziele (Use Cases)

Use Case	**Umsetzung**
Use Case 1: App starten	Umgesetzt
Use Case 2: Ball konfigurieren	Umgesetzt
Use Case 3: Login	Umgesetzt
Use Case 4: Registrieren	Umgesetzt
Use Case 5: Bestellung platzieren	Umgesetzt
Use Case 6: Bestellung abholen	Umgesetzt
Use Case 7: Daten über Seriennummer aufrufen	Umgesetzt
Use Case 8: Bestellliste	Umgesetzt
Use Case 9: Daten visualisieren	Umgesetzt
Use Case 10: Bestellungsfreigabe	Umgesetzt
Use Case 11: Benutzerdaten ändern	Nicht umgesetzt

Use Case 12: Sprache ändern	Nicht umgesetzt
Use Case 13: Push-Nachricht senden	Nicht umgesetzt

Tabelle 84: Implementierungsziele

8. Schlussbericht

8.1. Erfahrungsberichte

8.1.1. Simon Hager

"Je mehr ich weiss, um so mehr weiss ich, dass ich nicht(s) weiss." So lassen sich meine persönlichen Erfahrungen mit der Programmiersprache C# und dem Framework ASP.NET Core aus den vergangenen Wochen gut beschreiben. Der anfänglichen Naivität wich Frust, gefolgt von einem inneren Antrieb die Mechanismen hinter dem Toolset besser zu verstehen, um es für meine Bedürfnisse zielführend anzuwenden. Das hat jedoch seinen zeitlichen Tribut gefordert und einiges an Refactoring gebraucht. Ich denke, dass letzlich eine solide Lösung entstanden ist. Einige Aspekte sind jedoch sicher noch verbesserungswürdig. Zu nennen sind sicherlich das Testing. Die Mocking-Library wurden zu ausgiebig verwendet und das Mocking der Datenbank hat mich teilweise vor unüberwindbare Hindernisse gestellt. Zukünftig würde ich von Anfang an stärker auf eigene Mock-Konstsrukte zurückgreifen und wiederverwendeten Code stärker kapseln, um die Übersichtlichkeit einzelner Tests besser zu gewährleisten.

Auch kommunikativ hatte dieses Projekt seine Tücken. Anders als im EPJ, waren diesmal andere die Taktgeber. Da waren zum einen die St. Galler, die noch in Woche elf von Konzepten und vollständiger Integration ins SAP sprachen und Lukas Kretschmar, der lediglich einen Ball bestellen wollte. Das hat immer wieder zu grösseren Anpassungen geführt und grossen Einfluss auf den Projektverlauf genommen. Trotz dieser technologischen und kommunikativen Herausforderungen ist es uns gelungen, eine funktionierende Lösung zu entwickeln. Das freut mich sehr und zeigt mir, wie gut wir als Team funktioniert haben.

Was am Ende bleibt sind die neuen Erfahrungen und Fähigkeiten, welche durch die Auseinandersetzung mit den neuen Technologien und Stakeholdern entstanden sind. So konnte ich besonders viel praktisches Wissen in C# und ASP.NET Core, im CI/CD-Aufbau und dem Deployment mit Docker gewinnen.

8.1.2. Joel Schaltegger

Rückblickend würde ich das Projekt als Erfolg bezeichnen. Auch die Zusammenarbeit mit Lukas Kretschmar und meinen zwei Teamkollegen war stets angenehm. Bereits bei der Architekturplanung haben wir berücksichtigt, dass wir vielleicht eine Lösung ohne SAP umsetzen müssen. Dadurch war es einfacher, eine direkte Anbindung an die Smart Factory zu implementieren. Das Highlight während dem Projekt war für mich das Testen vom MVP. Dank Rafael Herzog, der kurzerhand die SPS Applikation im Techpark geschrieben hatte, konnten wir bei diesem Test den ersten Unihockeyball über unsere Bestell-App produzieren.

Da ich noch nie ein Softwareprojekt in ASP.NET umgesetzt habe, musste ich während der Implementation viel recherchieren. Dies führte dazu, dass es oft einen ersten Versuch der Lösung gab und ich es später neu schreiben musste, weil die Lösung an sich nicht gut umgesetzt war. Schwierigkeiten entstanden auch durch die Ungewissheit der SAP Schnittstellen. Wir haben versucht unser Backend auf das SAP abzustimmen und haben die Entities entsprechend aufgebaut. Weil jedoch nie bekannt wurde was für Daten das SAP vom Bestell-App braucht mussten wir viele Annahmen treffen.

8.1.3. Philipp Emmenegger

Die Studienarbeit empfand ich als eine positive Erfahrung. Ich bin bereits sehr motiviert in die Arbeit gestartet, da wir glücklicherweise unser Wunschthema erhalten haben. Die erste Herausforderung liess aber nicht lange auf sich warten. Die Planung des Projektes empfand ich als eine anspruchsvolle Aufgabe. Mir wurde bewusst, dass mir die Erfahrung in Softwareprojekten fehlte, um die Auswirkungen von Architekturentscheidungen richtig einschätzen zu können. Mittels Recherchearbeiten entwickelten wir trotzdem einen Plan, welcher mich zuversichtlich stimmte.

Sobald mit der Implementation des Projektes gestartet wurde, fühlte ich mich bereits etwas wohler. Meine Hauptaufgabe war das Realisieren vom Frontend der Bestell-App. Dank dem Engineering Projekt

hatte ich bereits etwas Erfahrung mit React, was mir dabei half, die Struktur der Webseite aufzubauen. Trotzdem begegnete ich vielen Herausforderungen, bei dessen Bewältigung ich aber enorm viel lernte. Insbesondere für das Lifecycle Management von React Komponenten und der Verwendung von Redux konnte ich mein Wissen stark ausbauen.

Ein weiteres Highlight der Studienarbeit war für mich die Zusammenarbeit im Team und mit dem Betreuer. Sowohl die Kommunikation, als auch die Stimmung waren immer erfreulich. Somit empfand ich das Arbeitsumfeld trotz manchmal unangenehmen, externen Einflüssen stehts als angenehm.

8.2. Danksagung

Im Verlauf der Semesterarbeit hatten wir die Gelegenheit mit einigen Mitarbeitern der OST zusammenzuarbeiten. Besonders positiv haben wir den Kontakt mit den Mitarbeitern im Techpark erlebt. Der Austausch war unkompliziert und unsere Zusammenarbeit führte stehts zu Erfolgen. Aus dem Techpark möchten wir uns deshalb besonders bei Rafael Herzog bedanken, der die Anbindung unserer Bestell-App an die Maschinensteuerung ermöglichte. Ausserdem bedanken wir uns bei Nicolas Wirz, der sich rasch in die Arbeit von Rafael Herzog eingearbeitet hat, um diese Anbindung zu erweitern.

Bei unserem Betreuer Lukas Kretschmar möchten wir uns speziell dafür bedanken, dass er sich immer schnell um unsere Anliegen kümmerte, uns bei wichtigen Entscheidungen gut beraten konnte und uns im Verlauf der Arbeit den Rücken frei gehalten hat. Auch der Austausch mit ihm war stets unkompliziert und angenehm. Das Vertrauen, welches er uns während der Arbeit schenkte, damit wir selbständig arbeiten konnten, schätzten wir sehr. Trotzdem unterstütze er uns mit kritischen Fragen, um unseren Fokus auf dem Wesentlichen zu halten.

8.3. Fazit

Ein wichtiger Aspekt zum Erfolg der Arbeit war die Zusammenarbeit und problemlose Arbeitsaufteilung im Team. Der interne Kommunikationsfluss war dank dem agilen Projektmanagement stehts gewährleistet. Dieses unterstütze uns ausserdem im Zeitmanagement, wodurch wir einen Zeitdruck beim Projektabschluss vermeiden konnten. Positiv wirkte sich auch unser Risikomanagement aus dem Projektplan aus. Beim Eintreten des Risikos bezüglich der SAP Schnittstellen konnten wir uns an dem dort definierten Verhalten orientieren. Durch die vorbeugenden Massnahmen wurde der dadurch generierte, zusätzliche Aufwand reduziert.

Die Planung der Architektur stellte sich als grössere Herausforderung dar. Aufgrund fehlender Erfahrungen waren wir gezwungen einige Recherchen zu betreiben, um weitsichtige Entscheidungen treffen zu können. Rückblickend konnten wir aber glücklicherweise feststellen, dass wir mit dem verwendeten Technologiestack zufrieden sind.

Beim Start in die Implementationsphase mussten wir unerwarted feststellen, dass die Kommunikation zwischen den Wirtschaftsinformatik Studierenden in St. Gallen und unserer Studienarbeit mangelhaft war. Die Zielsetzungen der beiden Arbeiten waren nicht darauf ausgelegt, parallel durchgeführt zu werden. Unsere Arbeit war auf die Produkte deren Seminararbeit angewiesen, welche für sie keine Priorität zu haben scheinten. Dieser Umstand wirkte sich negativ auf den weiteren Verlauf unserer Studienarbeit aus, da für uns über längere Zeit ungewiss war, ob wir benötigte Schnittstellen erhalten werden.

Mit der Entscheidung die Maschinensteuerung direkt anzubinden, stiessen wir auf neue Herausforderungen. Trotz allem gelang es uns eine eigene Bestellungsverwaltung zu entwickeln. In unserer Lösung greifen sowohl der Order System Proxy, als auch der Guerilla Container auf die selbe Datenbank zu. Der Order System Proxy hat dafür den Lead. Dies empfinden wir rückblickend als kleine Unschönheit in der Architektur und würden es beim nächsten Mal vermeiden.

Auch bei den Entities haben sich die fehlenden Informationen über das Bestellsystem SAP bemerkbar gemacht. Objekte wurden nur angenommen und zu lange dem fehlenden Bestellsystem ausgerichtet, sie

konnten nie mit der Zielarchitektur abgeglichen werden, was über den Projektverlauf in einigen Überbleibsel resultierte. Andererseits wurde dadurch angenommen, dass Bestellungen nicht selber gespeichert werden müssen, was zwischenzeitlich zu einem Schlüsselkonflikt geführt hat. Diese Probleme hätten durch eine stärker durchgesetzte Autonomie der Bestell-App mehrheitlich verhindert werden können.

Trotz einiger unangenehmer Umstände konnten wir die Motivation von Beginn der Arbeit bis zum Ende beibehalten. Mit dem Fokus auf Aspekte, die wir beeinflussen konnten, gelang es uns, eine erfolgreiche Studienarbeit abzuschliessen. Mit der ersten Bestellung eines Unihockeyballes über unser Bestell-App konnten wir das erhoffte Erfolgserlebnis geniessen und sogar ein Andenken daran mit nach Hause nehmen.

A. Verzeichnisse

A.1. Abkürzungen

Abkürzung	Bedeutung
ASP	Active Server Pages
CI/CD	Continous Integration / Continous Deployment
CORS	Cross-Origin Resource Sharing
DNS	Domain Name System
DoD	Definition of Done
DTO	Data transfer object
EF	Entity Framework
EPJ	Engineering Projekt
GraphQL	Graph Query Language
gRPC	Google Remote Procedure Call
IoT	Internet of things
JSON	JavaScript Object Notation
MVP	Minimum Viable Product
NFA	Non-Functional Requirements
PWA	Progressive Web Application
SAP	System Applications and Products
SASS	Syntactically Awesome Style Sheets
SF	Smart Factory
SPA	Single Page Application
SPS	Speicherprogrammierbare Steuerung
REST	Representational state transfer
RUP	Rational Unified Process
UC	Use Case
UI	User Interface
UX	User Experience
WINF	Wirtschaftsinformatik Studiengang

Tabelle 85: Abkürzungen

A.2. Glossar

Abkürzung	Bedeutung
AAAA	DNS Eintrag, der den Domain Namen auf eine IPv6 Adresse bindet.
ASP.NET	Ein Web Application Framework von Microsoft, mit dem sich dynamische Webseiten, Webanwendungen und Webservices.
Assemblies	Ausführbare Dateien (.exe) oder Dynamic Link Library-Dateien (.dll).
CNAME	Ein CNAME-Eintrag ist ein Art Alias für die DNS-Einträge einer anderen Domain.
CORS	Ein Mechanismus der es geschützten Ressourcen erlaubt von einer ander Domain angefordert zu werden.
DNS	Das Domain Name System und ist für die Namensauflösung eines Domainnamens in eine IP-Adresse zuständig.
DTO	Ein Transferobjekt bündelt mehrere Daten in einem Objekt, sodass sie durch einen einzigen Programmaufruf übertragen werden können.
EF Core	Ein Framework für objektrelationale Abbildung (ORM).
GraphQL	Eine Query Sprache für APIs. Bietet eine komplette und verständliche Beschreibung von Daten und gibt dem Client die Möglichkeit nach genau diesen Daten zu fragen welche er benötigt.
gRPC	Ein Remote Procedure Call framework um services effizient zu verbinden.
JSON	Ein kompaktes Datenformat in einer einfach lesbaren Textform für den Datenaustausch zwischen Anwendungen.
LaTeX	Anders als viele beim ersten Lesen denken, hat LaTeX nichts mit Kautschuk und Kondomen zu tun. Stattdessen handelt es sich um eine Software für Textverarbeitung und Formeln.
Let's Encrypt	Eine freie, automatisierte und offene Zertifizierungsstelle, herausgebracht für Sie durch Internet Security Research Group (ISRG).
Mendix	Eine Plattform mit niedrigem Code und hoher Produktivität, die es Unternehmen ermöglicht, ihre Innovations- und Wettbewerbsfähigkeit mit Anwendungen zu verbessern.
MindSphere	Ein vom Unternehmen Siemens entwickeltes offenes IoT-Betriebssystem. Es arbeitet cloudbasiert und ermöglicht es, Systeme, Maschinen, Anlagen und Produkte zu verbinden.
Proxy	Eine Kommunikationsschnittstelle in einem Netzwerk aus Rechnern. Er arbeitet als Vermittler, der auf der einen Seite Anfragen entgegennimmt, um dann über seine eigene Adresse eine Verbindung zur anderen Seite herzustellen.
PWA	Eine Progressive Web App oder PWA ist eine Anwendungssoftware, die Technologien wie HTML, CSS und JavaScript nutzt. Kurz gesagt sind PWAs Websites, die aussehen und funktionieren wie eine native App.
Redux	Eine quelloffene JavaScript-Bibliothek zur Verwaltung von Zustandsinformationen in einer Webanwendung.
Repositories	Zentrale Ablage in dem digitale Daten, Dokumente, Objekte und Programme mit ihren Metadaten verwaltet werden.

REST	Eine Programmierschnittstelle, die sich an den Paradigmen und Verhalten des World Wide Web (WWW) orientiert und einen Ansatz für die Kommunikation zwischen Client und Server in Netzwerken beschreibt.
Reverse Proxy	Ein Proxy in einem Rechnernetz, der Ressourcen für einen externen Client von einem oder mehreren internen Servern holt.
RUP	Ein bei IBM entwickeltes Vorgehensmodell für die Durchführung von Softwareprojekten.
SAP	Weltweit führende Software für die Steuerung von Geschäftsprozessen (ERP).
SASS	Eine Stylesheet-Sprache, die als CSS-Präprozessor mit Variablen, Schleifen und vielen anderen Funktionen, die Cascading Style Sheets (CSS) nicht mitbringen, die Erstellung von CSS vereinfacht und die Pflege grosser Stylesheets erleichtert.
Scrum	Die Scrum Methode ist ein Framework für agile Produktentwicklung und agiles Projektmanagement.
Sprint	Der wertschöpfende Projektprozess, bei dem das Entwicklungsteam innerhalb eines Vorgangs mit fixierter Dauer Anforderungen aus dem Sprint Backlog in ein Inkrement umsetzt.
SPS	Speicherprogrammierbare Steuerung, wird zur Steuerung und Regelung einer Maschine oder Anlage eingesetzt.
Technologie-Stack	Eine Liste aller Technologiedienste, die zum Aufbau und Ausführen einer einzelnen Anwendung verwendet werden.

Tabelle 86: Glossar

A.3. Abbildungsverzeichnis

A.4. Tabellenverzeichnis

Literatur

[1] R. Züllig, "Smart factory konfigurator," Bachelorarbeit, OST, FS 2021.

[2] "Iso 25000 portal." [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[3] "Google lighthouse." [Online]. Available: https://developers.google.com/web/tools/lighthouse

[4] "Siemens mendix." [Online]. Available: https://www.mendix.com

[5] "Comparing mongodb vs postgresql." [Online]. Available: https://www.mongodb.com/compare/mongodb-postgresql

[6] "Mindsphere api documentation." [Online]. Available: https://developer.mindsphere.io/apis/iot-iottimeseries/api-iottimeseries-api-swagger-3-6-0.html

[7] "Eslint." [Online]. Available: https://eslint.org

[8] "Sonarlint." [Online]. Available: https://www.sonarlint.org

[9] "Prettier." [Online]. Available: https://prettier.io

[10] "Stylecop." [Online]. Available: https://github.com/DotNetAnalyzers/StyleCopAnalyzers

[11] "Testing library mit jest." [Online]. Available: https://www.npmjs.com/package/@testing-library/jest-dom

[12] "xunit.net." [Online]. Available: https://xunit.net/

[13] "Fluent assertions." [Online]. Available: https://fluentassertions.com

[14] "Moq." [Online]. Available: https://github.com/moq/moq4

[15] "three.js." [Online]. Available: https://threejs.org

[16] "Tls challenge." [Online]. Available: https://letsencrypt.org/docs/challenge-types/#tls-alpn-01

[17] "Acme-protokoll." [Online]. Available: https://en.wikipedia.org/wiki/Automated_Certificate_Management_Environment

B. Anhang

B.1. Research

B.1.1. Mendix Research

Vorteile	Ist einfach in MindSphere zu integrieren
	Bietet CI/CD Support und Bereitstellung als Container
	Nutzt React-Native Framework, um einfach auf Android und IOS zu deployen
	Komponenten können selbst gecodet, oder per drag and drop erstellt werden
	REST-Schnittstellen für die MindSphere API sind vorhanden
Nachteile	Nur für Mobile-Apps ausgelegt, WebApps können mit Mendix nicht realisiert werden
	Mehr ein Baukasten mit vorgefertigten Modulen, als echtes Coden
	Externe Schnittstellen können vermutlich nicht genutzt werden

Tabelle 87: Mendix-Research

Mendix ist eine einfache Lösung, um industrielle Maschinen anzusteuern. Dies entspricht aber nicht den Anforderungen der Bestell-App. Für diesen Teil des Gesammtsystems ist das SAP zuständig. Die Bestell-App soll lediglich Produktionsdaten von MindSphere abrufen und diese abbilden.

B.1.2. MindSphere Research

Ziel von dieser Research ist es, herauszufinden, wie man die Produktionsdaten der Unihockeybälle über eine Schnittstelle abrufen kann. Siemens MindSphere bietet hier eine sehr umfängliche REST API. Um Daten aus dem Fleetmanager abzufragen kann der sogenannte IoT Time Series Service verwendet werden. In der Dokumentation der API ist von einer Base URL die Rede. Diese lautet in unserem Fall *gateway.eu1.mindsphere.io*. Möchte man den Request im Browser machen lautet die Base URL hsr-fleetmanager.eu1.mindsphere.io. Folgende Entities sind im MindSphere vorhanden:

Entity	**Entity-ID**	**PropertySets**
Battenfeld	22cc2a2298514c61b0dc96403b10233e	Environment, Machine, Product, Timestamp
WeldingMachine	05ba7fc079f04d6f85849312faf663dd	DateTime, Environment, Welding

Tabelle 88: Entities

Um die Datenmenge zu verringern gibt es die Möglichkeit bei einem Request die Parameter *from* und *to* mitzugeben. Das Format ist ein String im Standart ISO8601. Ein Request könnte am Ende also wie folgt aussehen:
https://hsr-fleetmanager.eu1.mindsphere.io/api/iottimeseries/v3/timeseries
/22cc2a2298514c61b0dc96403b10233e/Machine?
from=2021-08-01T12:00:00.000Z&to=2021-10-01T12:00:00.000Z

Schwierigkeiten entstehen, sobald man von einer externen Applikation auf die in MindSphere abgelegten Daten zugreiffen will. Das Problem hierbei ist es einen Bearer Token für die Authentifizierung zu generieren. Deshalb wird das Vorgehen im folgendem Unterkapitel beschrieben.

Bearer Token

Ziel	Beschreibung
Service Credentials für CloudFoundry erstellen	1. Vom Dashboard zu Settings navigieren 2. Den Tab Service Credentials öffnen und Service Credentials mit den Rollen *Organisationsmanager* und *hsr Entwickler* erstellen
CloudFoundry Direct URL auslesen	Um eine CloudFoundry URL zu erstellen, oder die vorhandenen anzusehen haben wir die CloudFoundry CLI verwendet. Diese ist unter folgender URL dokumentiert: *https://docs.cloudfoundry.org/cf-cli/getting-started.html* 1. CloudFoundry CLI installieren 2. Command *cf login* ausführen und mit den zuvor ertellten Service Credentials einloggen 3. Command *cf routes* ausführen und Host für die im nächsten Schritt zu erstellende App auswählen
Application erstellen	In unserem Fall ist die App eine API und die Infrastruktur ist MindShere Cloud Foundry. Als Cloud Foundry Direct URL haben wir *flow-server-hsr.apps.eu1.mindsphere.io* verwendet. 1. Vom Dashboard zum Developer Cockpit navigieren 2. Add application auswählen, Pflichtfelder ausfüllen und App erstellen
App Credentials erstellen	Hinweis: Diese Credentials sind 1 Jahr lang gültig und müssen dann neu erstellt werden. Die Bestell-App verwendet das daraus generierte Secret als Environment Variable, um einen Bearer Token zu generieren. 1. Im Developer Cockpit zu Tab Authorization Management navigieren 2. Unter App Credentials die vorhin erstellte App auswählen und Application Credentials erstellen 3. JSON File herunterladen
Access Token für MindSphere generieren	Hinweis: Optionaler Schritt für lokales testen - Die Bestell-App lösst bereits intern einen Access Token. 1. Postman installieren 2. Das JSON File ist eine Postman Collection und muss im Tool importiert werden 3. Den POST Request aus der importierten Collection absetzen 4. Als Response erhält man einen Access Token vom Type Bearer

Tabelle 89: Bearer Token

B.2. Responsive Screenshots

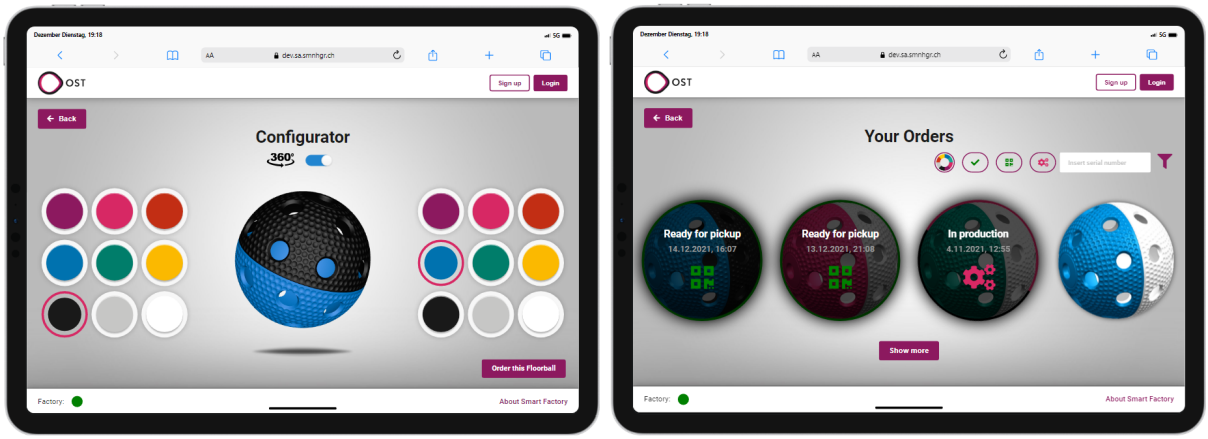

Abbildung 57: iPad 11 Pro (1194x834)

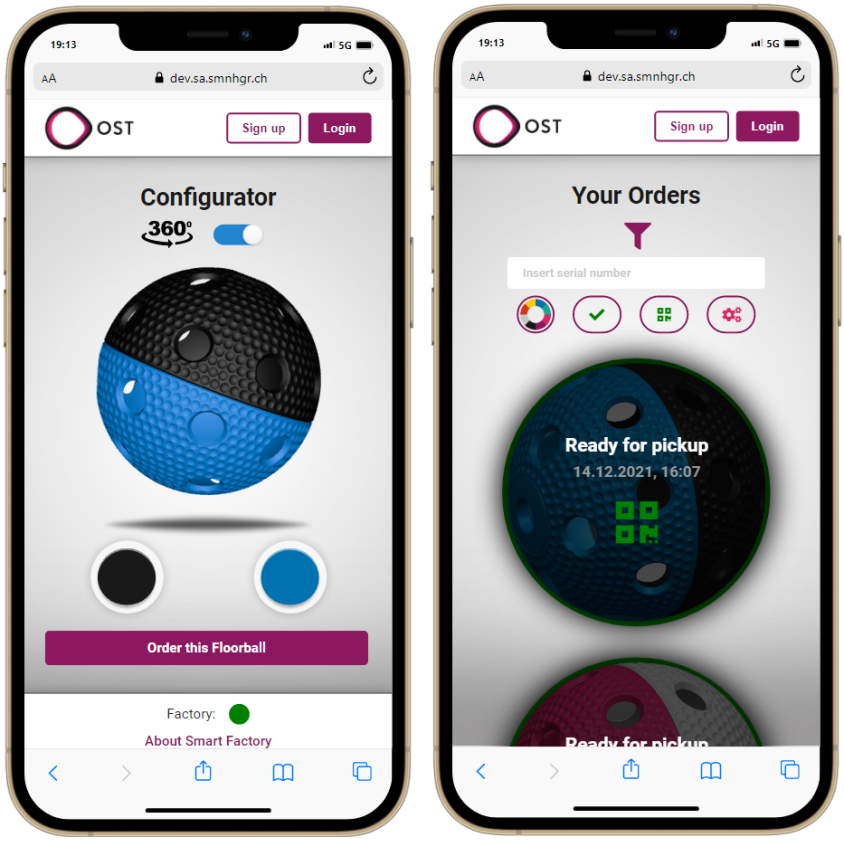

Abbildung 58: iPhone 12 Pro Max (428x926)

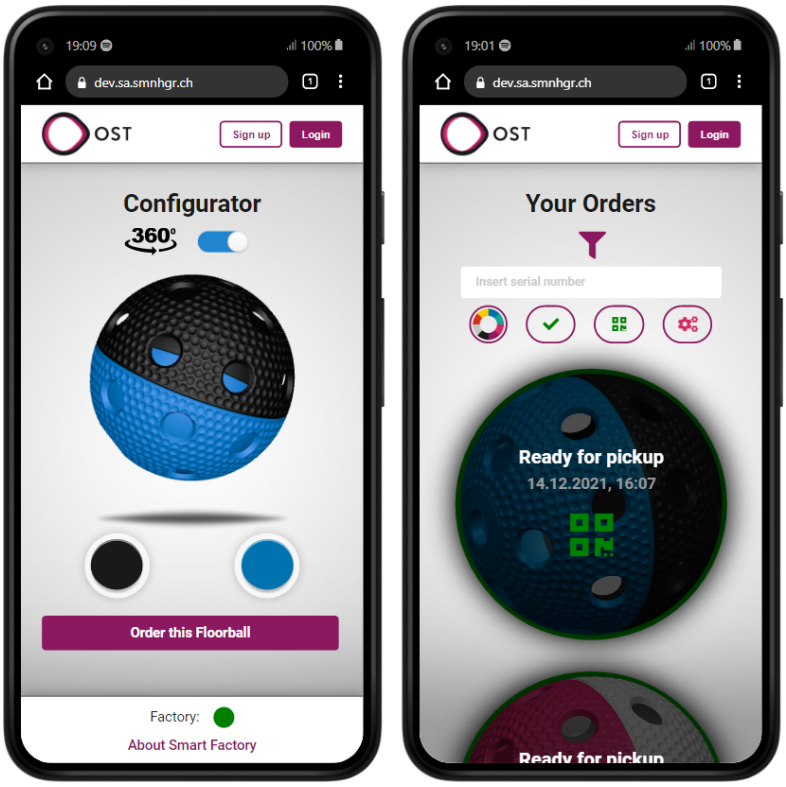

Abbildung 59: Google Pixel 5 (393x851)

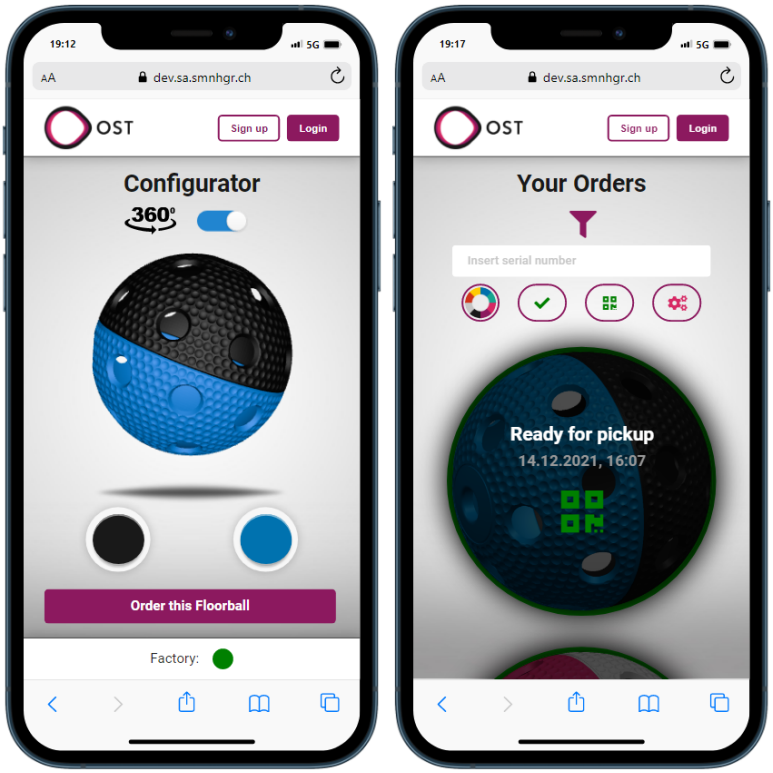

Abbildung 60: iPhone 12 Pro (390x844)

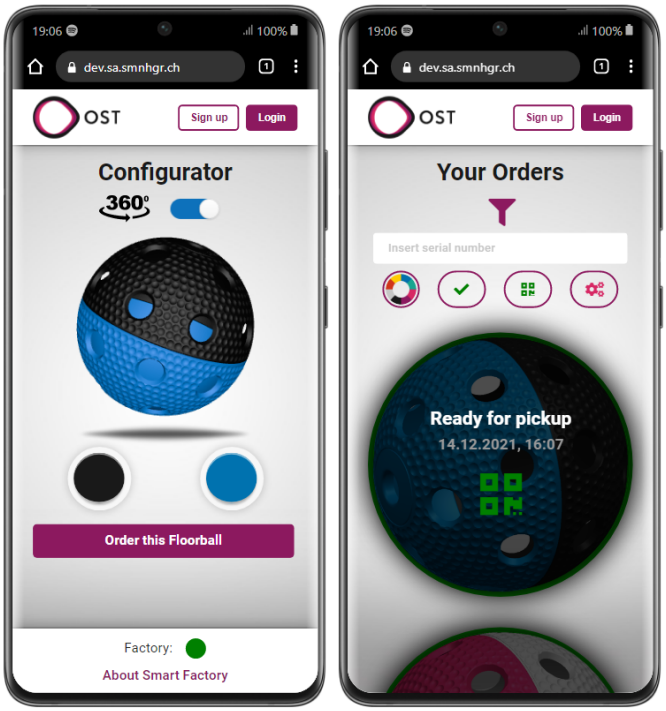

Abbildung 61: Samsung Galaxy S21 Ultra (360x800)

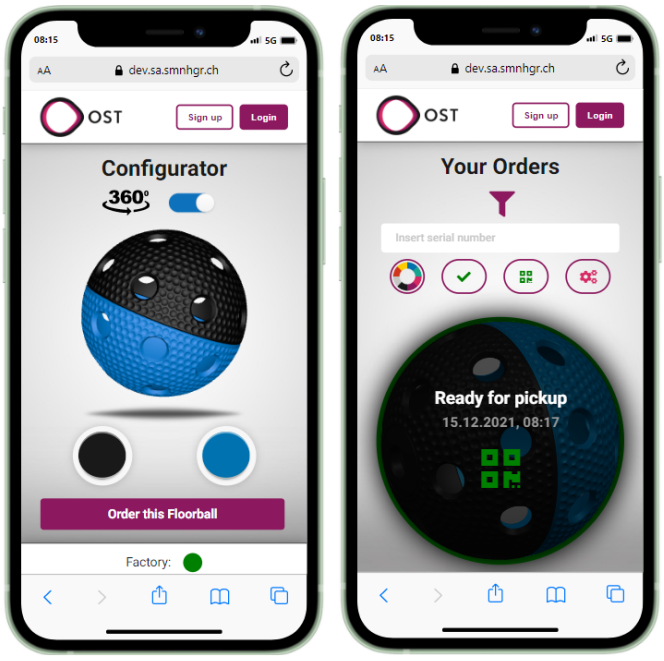

Abbildung 62: iPhone 11 Mini (360x780)

B.3. Testprotokolle

B.3.1. Sysemtest vom 14.12.2021

system_test.md 12/13/2021

System Test Protokoll

Vorbereitung

- [x] Aktuellste Versionen der Repositories sind deployed
- [x] Alle Container laufen
- [x] Web-App mit leerem Cache / leerem Local Storage geöffnet

Allgemein

- [x] OST-Logo wird dargestellt
- [x] Link auf Logo führt zu Home-Seite
- [x] Sign up / Login Buttons werden dargestellt und deren Link funktioniert
- [x] Factory Status im Footer wird dargestellt und ist korrekt

Nicht authentifiziert

Home

- [x] Titel wird dargestellt
- [x] Die drei Themen, sowie deren Beschreibung werden dargestellt
- [x] Die Links der Themen leiten auf die korrekte Seite weiter

Configurator

- [x] Bestell-Button ist deaktiviert und enthält Text: "Select two colors..."
- [x] Farbbestände werden korrekt dargestellt
- [x] Nicht verfügbare Farben können nicht ausgewählt werden
- [x] Wireframe des 2D-Balles wird dargestellt und dreht sich
- [x] Toggle zu 3D-Ball ist mit Label vorhanden

- Wähle eine Farbe aus

- [x] Bestell-Button ist deaktiviert und enthält Text: "Select another color..."
- [x] Eine Ballhälfte wird mit korrekter Farbe ersetzt
- [x] Ausgewählte Farbe wird Rahmen markiert

- Wechsle zur 3D-Ansicht

- [x] 3D Ball wird dargestellt
- [x] Rotations-Animation wird ausgeführt
- [x] Ausgewählte Farbe wird in die 3D-Ansicht übernommen
- [x] Ball kann manuell gedreht werden

- Wähle zweite Farbe aus

- [x] Bestell-Button ist aktiviert und enthält Text: "Order this Floorball"
- [x] Zweite Ballhälfte wird mit korrekter Farbe ersetzt

- Wechsle zur 2D-Ansicht

☑ Farbauswahl wird übernommen

- Bestelle konfigurierten Ball per klick auf Button

☑ Bestätigung wird verlangt

- Wähle Option: "No"

☑ Bestellung wird abgebrochen

- Bestelle konfigurierten Ball per klick auf Button
- Wähle Option: "Yes"

☑ Bestellbestätigung wird angezeigt
☑ Korrekte Farben werden im Bild dargestellt
☑ Bestelldatum, und Farben in der Beschreibung sind korrekt

- Klicke Button: "View your Orders"

☑ Weiterleidung auf Orders-Seite funktioniert

Orders

☑ Maximal 4 Bälle werden dargestellt
☑ Status der Bälle werden korrekt dargestellt
☑ Bei Statuswechsel werden Icon und Styles des Balles automatisch angepasst
☑ Sind weniger Bälle vorhanden wird kein Button dargestellt
☑ Sind mehr Bälle vorhanden wird der "Show more" Button dargestellt
☑ Bei Klick auf "Show more" Button werden jeweils 4 Bälle mehr dargestellt
☑ Bei Klick auf Filter Icon werden Filter eingeblendet

- Gebe eine Seriennummer ins Suchfeld ein

☑ Korreter Ball wird dargestellt

- Lösche Eingabe

☑ Alle Bälle werden angezeigt

- Filtere Nach Status

☑ Bälle mit korrektem Status werden angezeigt

- Setze Statusfilter zurück

☑ Alle Bälle werden angezeigt

- Filtere Nach Farbe

☑ Bälle mit ausgewählter Farbe werden angezeigt

- Setze Farbfilter zurück

- [x] Alle Bälle werden angezeigt

- Kombiniere mehrere Filter

- [x] Bälle die alle Filter erfüllen werden angezeigt

- Setze alle Filter zurück

- [x] Alle Bälle werden angezeigt

- Klicke auf einen Ball mit Status "Closed"

- [x] Detail-Modal öffnet sich
- [x] Icon und Informationen sind korrekt

- Klicke auf Button "Close"

- [x] Modal wird ausgeblendet

- Klicke auf einen Ball mit Status "inProduction"

- [x] Detail-Modal öffnet sich
- [x] Icon und Informationen sind korrekt

- Klicke neben Modal

- [x] Modal wird ausgeblendet

- Klicke auf einen Ball mit Status "readyForPickup"

- [x] Detail-Modal öffnet sich
- [x] QR-Code wird dargestellt
- [x] QR-Code enthält Seriennummer des Balles

- Klicke auf "Production Data" Button

- [x] Weiterleitung auf Production Data Seite funktioniert
- [x] Suche nach gewünschtem Ball wird ausgeführt

Production Data

- [x] Suchfeld mit Button wird dargestellt

- Suche nach fehlerhaften Seriennummer

- [x] Fehlermeldung erscheint

- Suche nach korrekter Seriennummer

- [x] Bilder des Korrekten Balles / Ballhälften werden dargestellt
- [x] MindSphere Daten werden dargestellt

Register

- [x] Link zu Login-Seite funktioniert

- Sende Formular leer ab

☑ Fehlermeldung "Invalid Firstname" wird angezeigt

- Fülle Feld "Firstname" aus
- Sende Formular ab

☑ Fehlermeldung "Invalid Lastname" wird angezeigt

- Fülle Feld "Lastname" aus
- Sende Formular ab

☑ Fehlermeldung "Invalid Email" wird angezeigt

- Fülle Feld "Email" aus
- Sende Formular ab

☑ Fehlermeldung zum Passwort wird angezeigt

- Fülle "Password" Feld aus
- Sende Formular ab

☑ Fehlermeldung "Passwords do not match" wird angezeigt

- Fülle "Confirm Password" Feld aus
- Sende Formular ab

☑ Weiterleitung zur Login Seite wird ausgeführt

Login

☑ Link zu Register-Seite funktioniert

- Gebe fehlerhafte Angaben ein

☑ Fehlermeldung wird angezeigt

- Gebe korrekte Angaben ein

☑ Login ist erfolgreich
☑ Weiterleitung auf Home-Seite wird ausgeführt
☑ Email und Logout Icon werden im Header angezeigt
☑ Access und Refresh Tokens werden im Local Storage abgelegt

Authentifiziert

Configurator

☑ Ball kann konfiguriert und bestellt werden
☑ Bei Bestellung wird im HTTP Request ein Authorization Header mit dem Access Token übermittelt

Orders

☑ Nur eigene Bälle werden angezeigt

Signout

- Klicke Signout-Icon im Header

☑ Weiterleitung auf Home-Seite funktioniert
☑ Im Header werden "Sign up" und "Login" Buttons angezeigt
☑ Access und Refresh Tokens wurden aus dem Local Storage entfernt
☑ Bei Requests werden keine Authorization Headers übermittelt

Admin

- Authentifiziere mit Admin Benutzer

☑ Anmeldung ist erfolgreich
☑ Admin Einstellungen im Footer werden angezeigt

- Navigieren auf die Configurator-Seite
- Deaktiviere Bestellungen

☑ Bestell Button wird mit deaktiviert-Meldung ersetzt

- Aktiviere Bestellungen

☑ Bestell Button wird wieder angezeigt

Angaben zur Durchführung

Datum: 14.12.2021

Verantwortlicher: Philipp Emmenegger

B.3.2. Sysemtest vom 21.12.2021

System Test Protokoll

Vorbereitung

- [x] Aktuellste Versionen der Repositories sind deployed
- [x] Alle Container laufen
- [x] Web-App mit leerem Cache / leerem Local Storage geöffnet

Allgemein

- [x] OST-Logo wird dargestellt
- [x] Link auf Logo führt zu Home-Seite
- [x] Sign up / Login Buttons werden dargestellt und deren Link funktioniert
- [x] Factory Status im Footer wird dargestellt und ist korrekt
- [x] About Smart Factory Link im Footer leitet auf About-Seite weiter
- [x] Bei ungültigen URLs wir die PageNotFound-Seite dargestellt

Nicht authentifiziert

Home

- [x] Titel wird dargestellt
- [x] Die drei Themen, sowie deren Beschreibung werden dargestellt
- [x] Die Links der Themen leiten auf die korrekte Seite weiter

Configurator

- [x] Bestell-Button ist deaktiviert und enthält Text: "Select two colors..."
- [x] Farbbestände werden korrekt dargestellt
- [x] Nicht verfügbare Farben können nicht ausgewählt werden
- [x] Wireframe des 2D-Balles wird dargestellt und dreht sich
- [x] Toggle zu 3D-Ball ist mit Label vorhanden

- Wähle eine Farbe aus

- [x] Bestell-Button ist deaktiviert und enthält Text: "Select another color..."
- [x] Eine Ballhälfte wird mit korrekter Farbe ersetzt
- [x] Ausgewählte Farbe wird Rahmen markiert

- Wechsle zur 3D-Ansicht

- [x] 3D Ball wird dargestellt
- [x] Rotations-Animation wird ausgeführt
- [x] Ausgewählte Farbe wird in die 3D-Ansicht übernommen
- [x] Ball kann manuell gedreht werden

- Wähle zweite Farbe aus

- Wechsle zur 2D-Ansicht

☑ Farbauswahl wird übernommen

- Bestelle konfigurierten Ball per klick auf Button

☑ Bestätigung wird verlangt

- Wähle Option: "No"

☑ Bestellung wird abgebrochen

- Bestelle konfigurierten Ball per klick auf Button
- Wähle Option: "Yes"

☑ Bestellbestätigung wird angezeigt
☑ Korrekte Farben werden im Bild dargestellt
☑ Bestelldatum, und Farben in der Beschreibung sind korrekt

- Klicke Button: "View your Orders"

☑ Weiterleidung auf Orders-Seite funktioniert

Orders

☑ Maximal 4 Bälle werden dargestellt
☑ Status der Bälle werden korrekt dargestellt
☑ Bei Statuswechsel werden Icon und Styles des Balles automatisch angepasst
☑ Sind weniger Bälle vorhanden wird kein Button dargestellt
☑ Sind mehr Bälle vorhanden wird der "Show more" Button dargestellt
☑ Bei Klick auf "Show more" Button werden jeweils 4 Bälle mehr dargestellt
☑ Bei Klick auf Filter Icon werden Filter eingeblendet

- Gebe eine Seriennummer ins Suchfeld ein

☑ Korreter Ball wird dargestellt

- Lösche Eingabe

☑ Alle Bälle werden angezeigt

- Filtere Nach Status

☑ Bälle mit korrektem Status werden angezeigt

- Setze Statusfilter zurück

☑ Alle Bälle werden angezeigt

- Filtere Nach Farbe

☑ Bälle mit ausgewählter Farbe werden angezeigt

- Setze Farbfilter zurück

- [x] Alle Bälle werden angezeigt

- Kombiniere mehrere Filter

- [x] Bälle die alle Filter erfüllen werden angezeigt

- Setze alle Filter zurück

- [x] Alle Bälle werden angezeigt

- Klicke auf einen Ball mit Status "Closed"

- [x] Detail-Modal öffnet sich
- [x] Icon und Informationen sind korrekt

- Klicke auf Button "Close"

- [x] Modal wird ausgeblendet

- Klicke auf einen Ball mit Status "inProduction"

- [x] Detail-Modal öffnet sich
- [x] Icon und Informationen sind korrekt

- Klicke neben Modal

- [x] Modal wird ausgeblendet

- Klicke auf einen Ball mit Status "readyForPickup"

- [x] Detail-Modal öffnet sich
- [x] QR-Code wird dargestellt
- [x] QR-Code enthält Seriennummer des Balles

- Klicke auf "Production Data" Button

- [x] Weiterleitung auf Production Data Seite funktioniert
- [x] Suche nach gewünschtem Ball wird ausgeführt

Production Data

- [x] Suchfeld mit Button wird dargestellt

- Suche nach fehlerhaften Seriennummer

- [x] Fehlermeldung erscheint

- Suche nach korrekter Seriennummer

- [x] Bilder des Korrekten Balles / Ballhälften werden dargestellt
- [x] MindSphere Daten werden dargestellt

Register

- [x] Link zu Login-Seite funktioniert

- Sende Formular leer ab

☑ Fehlermeldung "Invalid Firstname" wird angezeigt

- Fülle Feld "Firstname" aus
- Sende Formular ab

☑ Fehlermeldung "Invalid Lastname" wird angezeigt

- Fülle Feld "Lastname" aus
- Sende Formular ab

☑ Fehlermeldung "Invalid Email" wird angezeigt

- Fülle Feld "Email" aus
- Sende Formular ab

☑ Fehlermeldung zum Passwort wird angezeigt

- Fülle "Password" Feld aus
- Sende Formular ab

☑ Fehlermeldung "Passwords do not match" wird angezeigt

- Fülle "Confirm Password" Feld aus
- Sende Formular ab

☑ Weiterleitung zur Login Seite wird ausgeführt

Login

☑ Link zu Register-Seite funktioniert

- Gebe fehlerhafte Angaben ein

☑ Fehlermeldung wird angezeigt

- Gebe korrekte Angaben ein

☑ Login ist erfolgreich
☑ Weiterleitung auf Home-Seite wird ausgeführt
☑ Email und Logout Icon werden im Header angezeigt
☑ Access und Refresh Tokens werden im Local Storage abgelegt

Authentifiziert

Configurator

☑ Ball kann konfiguriert und bestellt werden
☑ Bei Bestellung wird im HTTP Request ein Authorization Header mit dem Access Token übermittelt

Orders

☑ Nur eigene Bälle werden angezeigt

Signout

- Klicke Signout-Icon im Header

☑ Weiterleitung auf Home-Seite funktioniert
☑ Im Header werden "Sign up" und "Login" Buttons angezeigt
☑ Access und Refresh Tokens wurden aus dem Local Storage entfernt
☑ Bei Requests werden keine Authorization Headers übermittelt

Admin

- Authentifiziere mit Admin Benutzer

☑ Anmeldung ist erfolgreich
☑ Admin Einstellungen im Footer werden angezeigt

- Navigieren auf die Configurator-Seite
- Deaktiviere Bestellungen

☑ Bestell Button wird mit deaktiviert-Meldung ersetzt

- Aktiviere Bestellungen

☑ Bestell Button wird wieder angezeigt

Angaben zur Durchführung

Datum: 14.12.2021

Verantwortlicher: Philipp Emmenegger