SYP-BSD-Projekt

Smart Gastro

Lotteritsch Andreas, Dizdarević Melisa & Mikula Elias

5AHIF, HTBLuVA Villach, SJ 2019/20

Table of Contents

[Projektauftrag 3](#_Toc30426903)

[Ausgangslage 3](#_Toc30426904)

[Auftraggeber 3](#_Toc30426905)

[Projektziele 3](#_Toc30426906)

[Ergebnis 3](#_Toc30426907)

[Technische Anforderungen und Rahmenbedingungen 0](#_Toc30426908)

[Projektorganisation und -planung 0](#_Toc30426909)

[Ernennung der Projektteams 0](#_Toc30426910)

[Sprint Information 0](#_Toc30426911)

[Zeitrahmen 1](#_Toc30426912)

[Lastenheft 2](#_Toc30426913)

[Allgemeines 2](#_Toc30426914)

[Ziel und Zweck des Dokuments 2](#_Toc30426915)

[Ausgangssituation 2](#_Toc30426916)

[Projektbezug 2](#_Toc30426917)

[Funktionale Anforderungen 2](#_Toc30426918)

[Design 2](#_Toc30426919)

[Ablauf 2](#_Toc30426920)

[Technische Umsetzung 2](#_Toc30426921)

[Akzeptanzkriterien 2](#_Toc30426922)

[Nichtfunktionale Anforderungen 3](#_Toc30426923)

[Allgemein 3](#_Toc30426924)

[Gesetzliche Anforderungen 3](#_Toc30426925)

[Technische Anforderungen 3](#_Toc30426926)

[Lieferumfang 3](#_Toc30426927)

[Ansprechstelle und Lieferort 3](#_Toc30426928)

[Abnahmevoraussetzungen 3](#_Toc30426929)

[Technelogie-Stack 4](#_Toc30426930)

[Iteration 1 4](#_Toc30426931)

[Iteration 2 5](#_Toc30426932)

[Iteration 3 6](#_Toc30426933)

[Klassendiagramm 7](#_Toc30426934)

[Konzeptionelles Modell 8](#_Toc30426935)

[Logisches Modell 9](#_Toc30426936)

[Use Case Diagramm 10](#_Toc30426937)

[User Stories 11](#_Toc30426938)

[Product-/Sprint-Backlog 15](#_Toc30426939)

[Iteration 1 15](#_Toc30426940)

[Iteration 2 15](#_Toc30426941)

[Changemanagement 16](#_Toc30426942)

[Daily Scrum Protokoll 17](#_Toc30426943)

[Arbeitsprotokolle 19](#_Toc30426944)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Projektauftrag | |  |  |
|  |  |  |  |  |
|  | **Projekt** | SmartGastro |  |  |
|  |  |  |  |  |
|  | **Ersteller** | Dizdarevic Melisa, Mikula Elias, Lotteritsch Andreas | Bildergebnis für htl villach |  |
|  |  |  |  |  |
|  | **Erstellt am** | 29.10.2019 |  |  |
|  | **Version** | 1.1 |  |  |
|  |  |  |  |  |
|  | **Hauptverantwortlicher**  **Qualitätssicherung** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Ausgangslage

Für ein Restaurant wird ein modernes Bestellsystem benötigt. Mittels Tablet sollen sich die Kunden allein ihre Bestellung zusammenstellen, welche automatisch zum Kellner und zur Küche gelangt und sofort bearbeitet werden kann, anstatt zu warten bis die Bestellung vom Kellner in die Küche gelangt.

### Auftraggeber

Der Auftraggeber ist das Restaurant.

### Projektziele

Ziel dieses Projektes ist es, den Kunden das Bestellen zu erleichtern und ihnen Wartezeit zu ersparen.

### Ergebnis

Das Ergebnis dieses Projektes sind eine Android App auf einem Tablet für die Kunden und Webpages für die Ansicht der Bestellungen für Kellner/Koch und eine Administrationswebpage für den Chef um z.B. neue Produkte hinzuzufügen, Preise zu ändern, etc.

## Technische Anforderungen und Rahmenbedingungen

Das zu entwickelnde Softwaresystem wird folgenden technischen Anforderungen und Rahmenbedingungen unterliegen:

* Frontend wird als Angular JS Webapp angezeigt
* Webservice zur Verarbeitung und Auswertung der Daten
* Android App, um Speisekarte einzusehen und Bestellungen zu tätigen

## Projektorganisation und -planung

Dieser Abschnitt beschreibt die Organisation und die Planungsvorgaben für das Projekt.

### Ernennung der Projektteams

|  |  |
| --- | --- |
| **Person** | **Rolle/Aufgaben/Fähigkeiten** |
| Fr. Dizdarevic | Entwicklung (Android), Organisatorisches |
|  |  |
| Hr. Mikula | Entwicklung (Webpage) |
|  |  |
| Hr. Lotteritsch | Entwicklung (Webservice) |
|  |  |

### Sprint Information

Sprint 1:

* Date: 28.10.2019
* Product Owner: Mikula Elias
* Scrum Master: Prof. Müller-Stegmüller

Sprint 2: (Iteration 1)

* 27.11.2019
* Product Owner: Lotteritsch Andreas
* Scrum Master: Prof. Müller-Stegmüller

Sprint 3: (Iteration 2)

* 22.01.2020
* Product Owner: Dizdarevic Melisa
* Scrum Master: Prof.Müller

### Zeitrahmen

* Projektstart: 27.09.2019
* Projektende: Anfang oder Ende März

# Lastenheft

|  |  |
| --- | --- |
| **Projektbezeichnung** | Tablet-Order-System for gastronomy |
| **Projektleiter** | Dizdarevic, Lotteritsch, Mikula |
| **Erstellt am** | 16.09.2019 |
| **Letzte Änderung am** | 16.09.2019 |
| **Status** | [in Bearbeitung] |
| **Aktuelle Version** | 1.0 |

## Allgemeines

### Ziel und Zweck des Dokuments

Dieses Lastenheft beschreibt die Anforderungen an das Maturaprojekt „Tablet-Order-System for gastronomy“.

Das Ziel ist ein Bestellsystem für die Gastronomie. Der Kunde/Die Kundin bestellt im Restaurant über ein Tablet und erspart sich so die Wartezeit bis der Kellner/die Kellnerin kommt. Die Bestellung geht dann direkt zum Computer der Bedienung.

### Ausgangssituation

Ein Restaurant benötigt eine Software, um die Bestellungen schneller und einfacher aufzunehmen.

### Projektbezug

Das vorliegende Projekt ist ein Gruppenprojekt im Rahmen des Unterrichts in der 5.ten Klasse im Schuljahr 2019/20.

## Funktionale Anforderungen

### Design

Das Design sollte sehr einfach zu bedienen sein und zudem eine gute Übersicht bringen. Die App soll sehr schnell und performant auf allen neueren Android Geräten einwandfrei laufen. Die monatlichen Kosten sollten sich auf ein Minimum beschränken.

### Ablauf

Der Kunde/Die Kundin bestellt per Tablet. Die Bestellung gelangt danach sofort zum Computer der Bedienung, der die Getränke übernimmt und die Gerichte werden auf einem Bildschirm in der Küche angezeigt, so weiß man sofort, was zu kochen ist.

### Technische Umsetzung

Die technische Umsetzung erfolgt in drei Phasen, welche mit genaueren Details im Dokument [Idee BSD 5A](file:///C:\Users\Misa\Desktop\IdeeBSD%205A.docx) zu finden sind.

### Akzeptanzkriterien

MUSS:

* Bestellung immer nur auf einem Tisch
* Ein Tablet eine Rechnung
* Posten aus Bestellung auf der Webpage wird vom Kellner gelöscht, wenn gebracht

SOLL:

* Kellner wählt sich am Client aus einer Liste aus 🡪 Kellner zugeordnet

KANN:

* Authentifizierungssystem für Kellner

## Nichtfunktionale Anforderungen

### Allgemein

Allgemeine Anforderungen des Projekts liegen darin, dass in einer Gastronomie die Mitarbeiter und Mitarbeiterinnen eine App benutzen, um Bestellungen von Kunden zu verwalten. Die Benützung sollte intuitiv sein und natürlich schnell. In der Küche sollten die Bestellungen dann via Bildschirm angezeigt werden.

### Gesetzliche Anforderungen

Zu den Gesetzlichen Anforderungen ist der Datenschutz gemäß dem DSVGO zu beachten und sollte umgehend eingehalten werden, um die Privatsphäre von Kunden zu schützen

### Technische Anforderungen

Die App sollte auf jedem neueren Android Smartphones flüssig laufen. Die Entwicklung sollte vorzugsweise in Android Studio und in Java stattfinden. Die Daten sollten nur intern im lokalen Netzwerk ausgetauscht werden und somit nicht ins Internet gelangen. In der Küche sollten dann alle Bestellungen via einer Weboberfläche angezeigt werden.

## Lieferumfang

Ein Softwaresystem für Restaurants, welche die drahtlose, vollautomatische Bestellung eines Gastes und Übermittlung der Bestellung in die Küche über ein Tablet ermöglicht.

### Ansprechstelle und Lieferort

HTBLuVA Villach

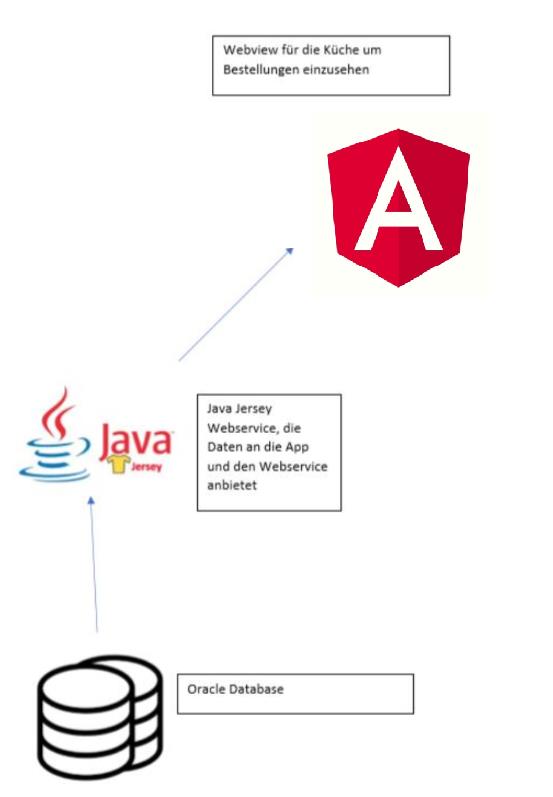
## Abnahmevoraussetzungen

Das Projekt gilt als abgeschlossen, wenn das System fertiggestellt ist, und der automatisierte Bestellungsprozess für alle Anwender reibungslos verläuft.

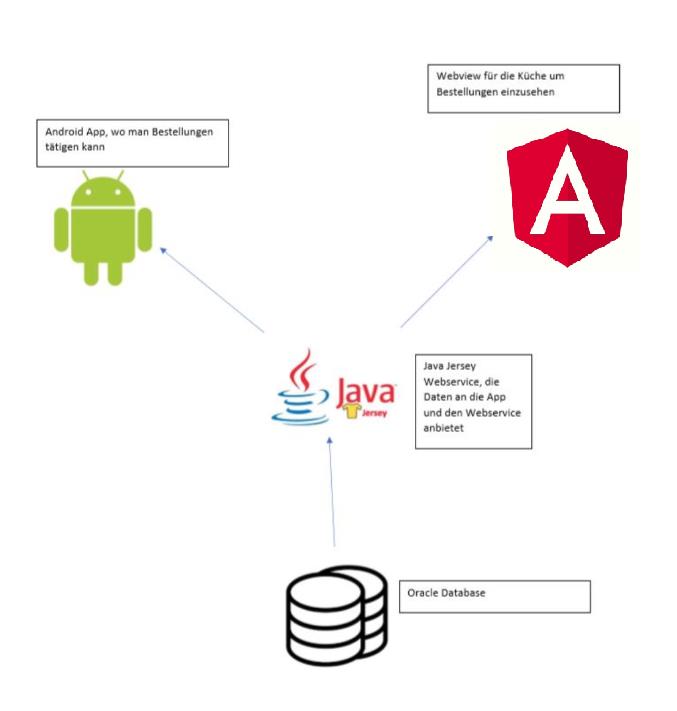
Die Professoren Linder, Reichmann und Müller-Stegmüller definieren, ob die Qualität des Projektes stimmt.

# Technelogie-Stack

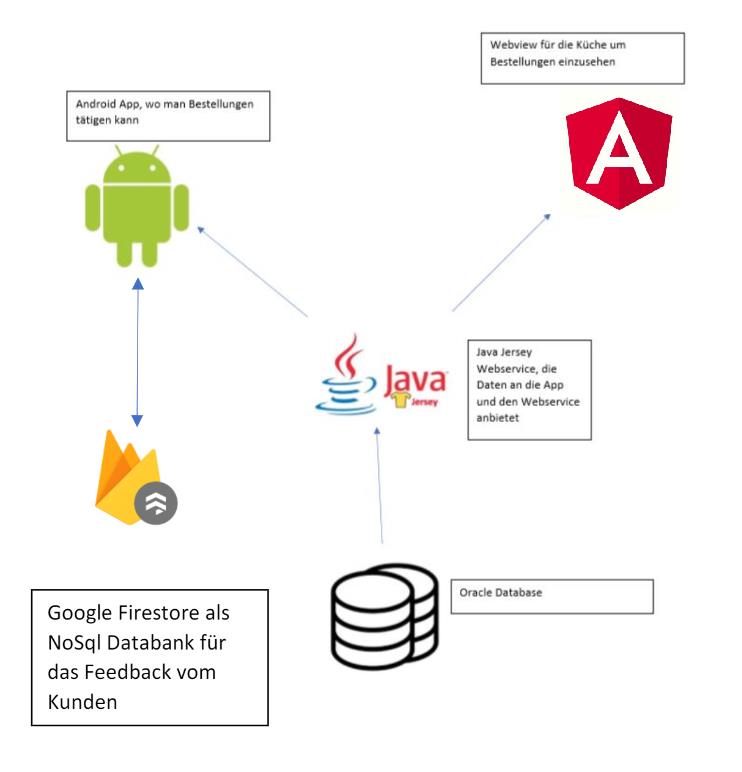
## Iteration 1



## Iteration 2



## Iteration 3



# Klassendiagramm

A screenshot of a cell phone

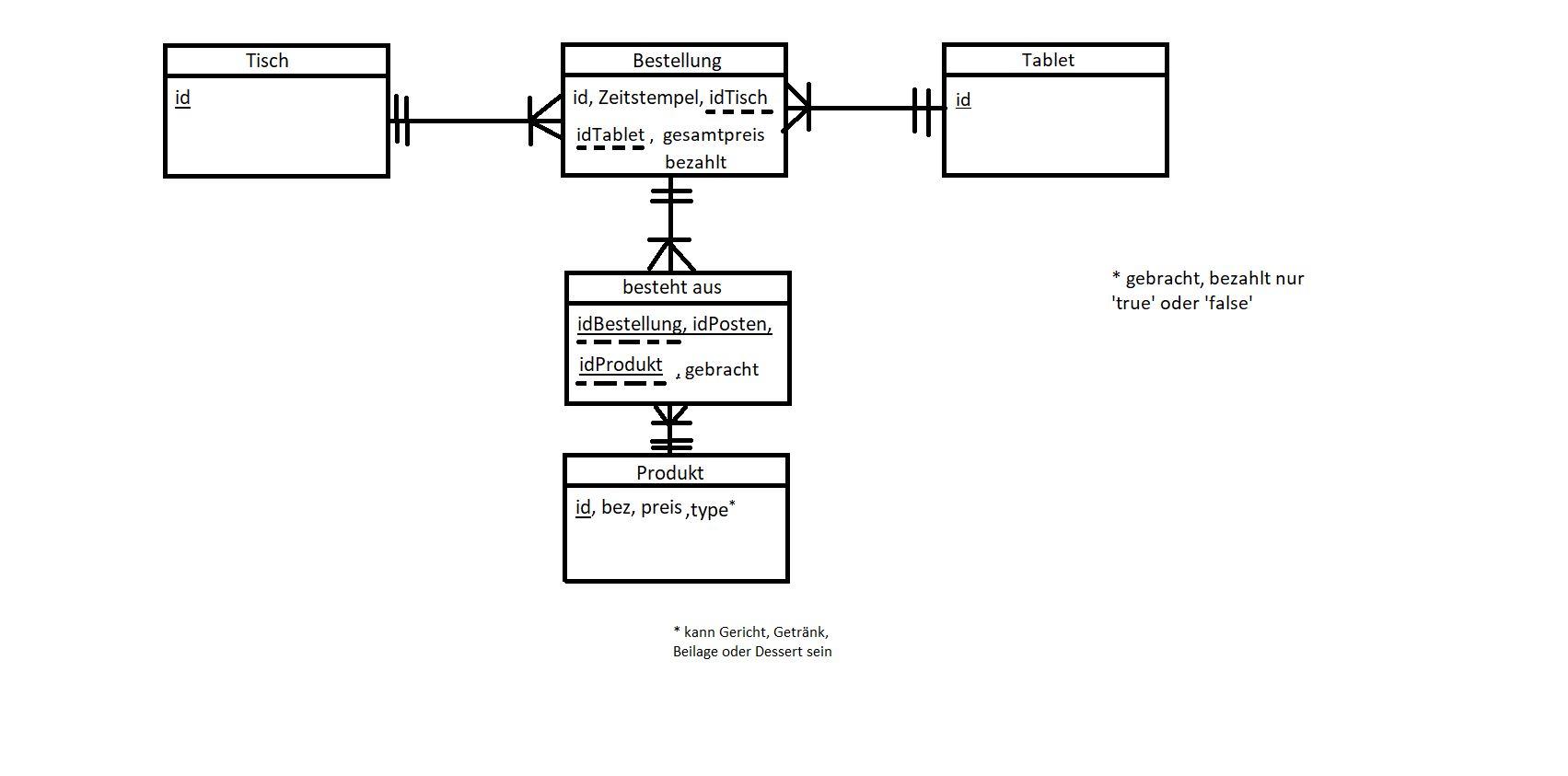
Description automatically generated

# Konzeptionelles Modell

A screenshot of a cell phone

Description automatically generated

# Logisches Modell



# Use Case Diagramm

A close up of a map

Description automatically generated

# User Stories

1. User Story:

|  |  |  |
| --- | --- | --- |
| Rolle: | User Story: | Grund: |
| Als Kunde möchte ich | die Karten ansehen können | Essen bestellen |

Akzeptanzkriterien:

* Tisch verfügbar, Tablet bei Tisch
* Tablet verfügbar und mit Server verbunden

Testfälle:

1. Ok

* Vorbedingung:
  + User muss Tablet bekommen und es entsperren
* Aktion:
  + User drückt auf Menüs Speise, Getränk, Beilage, Dessert
* Ergebnis:
  + Der User sieht nun immer die Seite mit den jeweiligen Produkten

1. Nicht ok:

* Vorbedingung:
  + App nicht mit Server verbunden bzw. Server down
* Aktion:
  + Der User versucht eine Karte zu öffnen
* Ergebnis:
  + Die Karte wird nicht geladen, angezeigt

1. User Story:

|  |  |  |
| --- | --- | --- |
| Rolle: | User Story: | Grund: |
| Als Kunde möchte ich | eine Speise bestellen | hungrig |

Akzeptanzkriterien:

* Tablet verfügbar und mit Server verbunden
* Menüauswahl wird angezeigt

Testfälle:

1. Ok

* Vorbedingung:
  + User muss Tablet bekommen und es einschalten
  + User drückt auf Speisekarte
* Aktion: User wählt eine Speise aus, z.B. Schnitzel (legt es in Warenkorb) und drückt auf Button Bestellen
* Ergebnis:
  + Der User hat nun das Schnitzel bestellt und es ist auf der Webpage in der Küche und beim Kellner angelangt

1. Nicht ok:

* Vorbedingung:
  + App nicht mit Server verbunden bzw. Server down
* Aktion:
  + Der User versucht Schnitzel in Warenkorb zu legen
* Ergebnis:
  + Warenkorb stürzt ab

1. User Story:

|  |  |  |
| --- | --- | --- |
| Rolle: | User Story: | Grund: |
| Als Kunde möchte ich | ein Getränk bestellen | durstig |

Akzeptanzkriterien:

* Tablet verfügbar und mit Server verbunden
* Menüauswahl wird angezeigt

Testfälle:

1. Ok

* Vorbedingung:
  + User muss Tablet bekommen und es einschalten
  + User drückt auf Getränkekarte
* Aktion: User wählt eine Getränk aus, z.B. Coca Cola (legt es in Warenkorb) und drückt auf Button Bestellen
* Ergebnis:
  + Der User hat nun die Cola bestellt und es ist auf der Webpage beim Kellner angelangt

1. Nicht ok:

* Vorbedingung:
  + App nicht mit Server verbunden bzw. Server down
* Aktion:
  + Der User versucht Cola in Warenkorb zu legen
* Ergebnis:
  + Warenkorb stürzt ab

1. User Story:

|  |  |  |
| --- | --- | --- |
| Rolle: | User Story: | Grund: |
| Als Kunde möchte ich | eine Beilage bestellen | möchte Schnitzel mit Pommes |

Akzeptanzkriterien:

* Tablet verfügbar und mit Server verbunden
* Menüauswahl wird angezeigt

Testfälle:

1. Ok

* Vorbedingung:
  + User muss Tablet bekommen und es einschalten
  + User drückt auf Beilagenkarte
* Aktion: User wählt eine Beilage aus, z.B. Pommes (legt es in Warenkorb) und drückt auf Button Bestellen
* Ergebnis:
  + Der User hat nun die Pommes bestellt und es ist auf der Webpage in der Küche und beim Kellner angelangt

1. Nicht ok:

* Vorbedingung:
  + App nicht mit Server verbunden bzw. Server down
* Aktion:
  + Der User versucht Pommes in Warenkorb zu legen
* Ergebnis:
  + Warenkorb stürzt ab

1. User Story:

|  |  |  |
| --- | --- | --- |
| Rolle: | User Story: | Grund: |
| Als Kunde möchte ich | ein Dessert bestellen | möchte Tiramisu essen |

Akzeptanzkriterien:

* Tablet verfügbar und mit Server verbunden
* Menüauswahl wird angezeigt

Testfälle:

1. Ok

* Vorbedingung:
  + User muss Tablet bekommen und es einschalten
  + User drückt auf Dessertkarte
* Aktion: User wählt ein Dessert aus, z.B. Tiramisu (legt es in Warenkorb) und drückt auf Button Bestellen
* Ergebnis:
  + Der User hat nun das Tiramisu bestellt und es ist auf der Webpage in der Küche und beim Kellner angelangt

1. Nicht ok:

* Vorbedingung:
  + App nicht mit Server verbunden bzw. Server down
* Aktion:
  + Der User versucht Tiramiss in Warenkorb zu legen
* Ergebnis:
  + Warenkorb stürzt ab

1. User Story:

|  |  |  |
| --- | --- | --- |
| Rolle: | User Story: | Grund: |
| Als Kellner möchte ich | mich aus der Kellnerliste für eine Bestellung auswählen | Welcher Kellner welche Bestellung bearbeitet |

Akzeptanzkriterien:

* Webseite verfügbar und mit Server verbunden
* Computer verfügbar

Testfälle:

1. Ok

* Vorbedingung:
  + User muss Webseite aufrufen
  + User drückt auf Bestellung
* Aktion: User drückt auf Kellnerliste und wählt seinen Namen aus
* Ergebnis:
  + Der User ist nun für die ausgewählte Bestellung verantwortlich

1. Nicht ok:

* Vorbedingung:
  + App nicht mit Server verbunden bzw. Server down
* Aktion:
  + Der User versucht sich auszuwählen
* Ergebnis:
  + Die Liste ist leer

1. User Story:

|  |  |  |
| --- | --- | --- |
| Rolle: | User Story: | Grund: |
| Als Kellner möchte ich | einen Posten einer Bestellung löschen | Habe das Produkt zum Tisch gebracht und will es abhaken |

Akzeptanzkriterien:

* Webseite verfügbar und mit Server verbunden
* Computer verfügbar

Testfälle:

1. Ok

* Vorbedingung:
  + User muss Webseite aufrufen
  + Aktion: User drückt auf Haken(Combobox) neben jeden Posten
* Ergebnis:
  + Der Posten erlischt nun und nur mehr die restlichen Sachen, die zu bringen sind, erscheinen

1. Nicht ok:

* Vorbedingung:
  + App nicht mit Server verbunden bzw. Server down
* Aktion:
  + Der User versucht die Seite aufzurufen
* Ergebnis:
  + Die Seite ist leer

# Product-/Sprint-Backlog

## A screenshot of a cell phone Description automatically generatedIteration 1

## Iteration 2

# Changemanagement

**09.12.2019**

Anstatt in der Datenbank auf die noch zu zahlenden Bestellungen über die Uhrzeit zuzugreifen, legen wir ein Bezahlt-Flag und für die Kellner/Küche ein Gebracht-Flag.

Der Kunde bestellt über die Tablet-App und schickt die Bestellung ab. Die Bestellung enthält eine Bestellnummer und die jeweilige Tablet-ID. Das Bezahlt-Flag und Gebracht-Flag sind derzeit noch auf false. In den Webapps für Kellner/Küche werden die neuen Bestellungen angezeigt(also die, wo das Gebracht-Flag auf false ist).

Ist die Bestellung zu Tisch gebracht, kann der Kellner mit einem Klick auf die Bestellung das Gebracht-Flag auf true setzen und dieses erlischt von der Oberfläche.

Will der Kunde noch etwas essen, so wird dies als neue Bestellung getätigt(z.B. ein Dessert). Es ist eine neue Bestellung, aber logischerweise noch die gleiche Tablet-ID.

Um am Ende die Bezahlung zu ermöglichen, wird das Tablet zum Kellner gebracht und dieser wertet mithilfe einer Abfrage aller Bestellungen vom Tablet xy, die noch nicht bezahlt wurden (also wo das Bezahlt-Flag) noch nicht gesetzt wurde. Nach dem Bezahlvorgang wird dieses gesetzt und nun kann wieder wer anders das Tablet xy bekommen.

# Daily Scrum Protokoll

|  |  |
| --- | --- |
| **Datum** | **Protokoll** |
| 30.09.19 | Wir arbeiten weiterhin an einem Datenmodell, welches unseren Vorstellungen entspricht - zurzeit sind wir für die Lösung mit einem Flag - jedes Produkt enthält einen Flag, ob es nun ein Gericht, eine Beilage, ein Dessert, oder ein Getränk ist |
| 07.10.19 | Dizdarević stellt das ER Diagramm fertig, Lotteritsch schreibt die Grundstruktur des WebServices, und Mikula erkundigt sich über das Frontend und mögliche Technologien |
| 14.10.19 | Dizdarević arbeitet am Scrum-Board, Lotteritsch macht das Use-Case Diagram, und Mikula arbeitet an Entwürfen für die Iterations-Grafiken |
| 21.10.19 | Lotteritsch stellt heute das Use-Case Diagramm fertig, Dizdarevic arbeitet an der ersten physischen Implementierung für die Datenbank, und Mikula stellt die vorläufigen Iterations-Grafiken fertig |
| 28.10.19 | Heute ist Sprint Review. Soweit läuft alles gut. Allerdings haben wir ein Problem mit der Entwicklung der Datenbank - wir finden kein vernünftiges Modell, mit dem wir die Bestellungen abhaken und bezahlen lassen können - wir sind also am überlegen, wie wir es am besten lösen werden |
| 04.11.19 | Heute konzentrieren wir uns darauf, bald die erste Funktionalität austesten zu können - Lotteritsch arbeitet am WebService weiter, Dizdarević entwirft weiterhin die Datenbank, und Mikula arbeitet an den ersten Frontend-Entwürfen. Das Problem mit dem Datenbankmodell ist noch nicht vollständig gelöst |
| 11.11.19 | Heute programmieren wir weiter - Dizdarević an der Datenbank, Lotteritsch am WebService, und Mikula am Frontend, welches er mit Angular lösen möchte. Das Problem mit dem Datenbankmodell besteht weiter, wir reden über den nächsten Lösungsansatz, sobald es für die Programmierung wieder relevant wird, um Ideen zu sammeln |
| 18.11.19 | Der erste große Erfolg - Der Client verbindet holt sich Daten vom WebService, welches die Daten wiederum aus der Datenbank lädt. Anschließend zeigt der Client die Daten dynamisch als Bestellungen im Frontend an |
| 25.11.19 | Lotteritsch wird heute neue Routen für das WebService implementieren, Mikula arbeitet an der Weboberfläche, und Dizdarević kümmert sich um diverse Protokolle |
| 02.12.19 | Lotteritsch arbeitet weiter am WebService, Mikula auch weiter an der Weboberfläche, und Dizdarević kümmert sich schon um die Android App |
| 09.12.19 | Mikula stellt heute die erste Weboberfläche ziemlich fertig, mit von mir bereitgestellter Testdaten. Die dazugehörigen Routen werden von Lotteritsch heute auch fertiggestellt. Es geht um Routen für Gruppierte Bestellungen, jeweils für den Kellner und die Küche. Dizdarević arbeitet weiter an der Android App |
| 16.12.19 | Das Gesamtpaket mit dem Übermitteln von Gruppierten Bestellungen an das WebService soll heute funktionieren - jeder kümmert sich um seinen Teil, sodass die Verbindung heute noch funktioniert |
| 23.12.19 | Weihnachtsferien |
| 30.12.19 | Weihnachtsferien |
| 06.01.20 | Weihnachtsferien |
| 13.01.20 | Dizdarević hat die Grundfunktionalitäten für die Android-App fertiggestellt, nun muss Lotteritsch weitere Routen entwickeln, damit die Android-App Bestellungen abschicken kann - Lotteritsch & Mikula arbeiten zudem an der erfolgreichen Übermittlung von Requests an das Webservice mit Berücksichtigung auf CORS |
| 20.01.20 | Heute schreiben wir alle ein Master Dokument mit allen Prozessbegleitenden Dokumenten |

# Arbeitsprotokolle

|  |  |  |  |
| --- | --- | --- | --- |
|  | Lotteritsch | Mikula | Dizdarevic |
| **30.09.19** | Gemeinsames arbeiten am Datenmodell | Gemeinsames arbeiten am Datenmodell | Gemeinsames arbeiten am Datenmodell |
| **07.10.19** | Erstellung des Webservice, überlegung und Aufbau der Grundstruktur | Research über Frontend Technologien und über deren möglichen Anwendungen | Fertigstellung des ER Diagramm, für spätere Implementierung der Datenbank |
| **14.10.19** | Erstellung Use-Case Diagramm | Erstellung der Iterations-Grafiken | Erstellung eines SCRUM-Board |
| **21.10.19** | Fertigstellung des Use-Case Diagramm | Fertigstellung der Intertions-Grafiken | Erstellung und Implementierung der Datenbank |
| **28.10.19** | Überarbeitung des Datenbankmodelles | Überarbeitung des Datenbankmodelles | Überarbeitung des Datenbankmodelles |
| **04.11.19** | Fertigstellung weiterer Routen im Webservice (get /meals) | Arbeiten an ersten Frontend-Entwürfen | Überarbeitung des Datenbankmodelles |
| **11.11.19** | Erstellung weitere Routen im Webservice | Erstellung der ersten Web-App mithilfe von Angular | Erstellung von Testdaten in der Datenbank für das Webserivce |
| **18.11.19** | Fixing Bugs und Zusammenarbeit mit Mikula um Verbindung zwischen Webservice und WebApp zu ermöglichen | Fertigstellung des HTTP-Serivce im Frontend um Daten vom Webservice zu bekommen | Weitere Aufbereitung der Testdaten, sowie überlegung von anderen Implementierung der Datenbank |
| **25.11.19** | Weitere implementierung von neuen Routen im Webservice | Arbeiten an der Weboberfläche | Erstellung von diveren Protokollen |
| **02.12.19** | Implementierung von neuen Routen im Webservice | Weiterarbeiten an der Weoberfläche damit die Daten für den Nutzer übersichtlicher angezeigt werden | Erstellung der Android App |
| **09.12.19** | Fertigstellung der Route "Gruppierte Bestellungen" | Grobe Fertigstellung der Weboberfläche | Weiterarbeiten an der Android App |
| **16.12.19** | Weiterarbeiten am Webservice | Fixing Bugs in der WebApp | Arbeiten an der Android App, erstellung des Bestellmenüs in der Android App |
| **13.01.20** | Zusammenarbeit mit Mikula um Daten an die WebApp zu übermitteln, CORS macht probleme | Zusammenarbeit mit Lotteritsch um Daten an die WebApp zu übermitteln, CORS macht probleme | Fertigstellung der Grundfunktionalität der Android App |
| **20.01.20** | Schreiben des Master Dokumentes | Schreiben des Master Dokumentes | Schreiben des Master Dokumentes |