**Döner-Trainer**

Die Furtwanger Döner-Buden bereiten sich auf die Post-Corona-Zeit vor und möchten für den erwarteten Andrang ihre Prozesse optimieren. So haben sie sich zusammengeschlossen und sich an die Fakultät ihres Vertrauens in der direkten Nachbarschaft gewandt, um sich eine kleine Simulation bauen zu lassen, mit der sie ein paar Zusammenhänge erforschen können. Tabellenkalkulationen sind langweilig, sie möchten die Erfahrungen eher in visuell und ggf. akustisch ansprechender Form machen. Es wurde folgendes vereinbart:

Dargestellt werden soll die Theke, wo die Döner, Yufka und Lahmacun gebaut werden, mit den Behältnissen für die Zutaten, der hintere Teil der Küche, wo das Rohmaterial liegt, verarbeitet wird und damit die Behältnisse aufgefüllt werden, und der Raum vor der Theke, wo die Kundis ihre Bestellungen abgeben, bezahlen und das Produkt entgegennehmen.

* UI SCRIBBLE

Mitarbeiteris und Kundis werden schematisch dargestellt, legen Wege zurück, transportieren Container etc.

* Bewegung auf Canvas. Mitarbeiter / Kunden als Smileys
* Mitarbeiter: Küche, Kunden: Theke

Die Mengen der Füllstände werden an den Containern und dem Rohmaterial angezeigt, so dass das Nutzeri stets einen Überblick hat.

* Progress Bar (vielleicht unübersichtlich), Mit Hover erkennbar

Der Zustand der Akteure in der Simulation ist auch erkennbar, ob bsp. ein **Kundi wütend** ist oder das **Personal schläft oder total gestresst** ist ...

* Unterscheidung Kunde und Mitarbeiter. Kunde: Entspannt, ungeduldig, wütend. Mitarbeiter: Konzentriert, müde, gestresst
* Stimmung anhand von Smiley darstellen, anhand deren die Stimmung abgelesen werden kann

Das Nutzeri betreibt die Dönerbude. Es **schickt das Personal** herum, damit nicht den ganzen Tag dieselbe Person an der gleichen Station ist, und sorgt für **einzelne Handlungen auf hohem Detailgrad**, damit beispielsweise Kraut auf dem Yufka landet, oder der **Zwiebelcontainer gefüllt** wird und schließlich mit dem ziemlich leeren Container in der Theke **ausgetauscht** wird. Weiterhin muss es dafür sorgen, dass rechtzeitig Rohmaterial **nachbestellt** wird, damit es nicht ausgeht.

* Aufgabe vom Nutzer:
  + Personal herumschicken
  + Yufkas, Döner selbst bauen?
  + Container füllen und mit dem der Theke austauschen
  + Rohmaterial nachbestellen

Manche Prozesse wie das Zwiebelschneiden oder die Anlieferung des Rohmaterials brauchen dabei Zeit und sind nicht sofort erledigt.

* Time Events. Animationen, nicht sofort darstellen lassen (Im Bereich der Küche)

An der Theke geht aber alles recht fix, da landet das Kraut ziemlich bald mit der Interaktion auf dem Yufka.

* Drag and Drop? Click Event?

Die Kundis kommen in den Laden und teilen ihre Wünsche mit, wie "keine Zwiebeln" oder "extra scharf",

* Jeder Kunde tätigt individuelle Bestellung. Random.
* Bestellung = Hähnchen/Kalb/Falafel + Döner/Yufka/Lahmacun + Zutaten: Mit allem oder ohne Zutat XY + Extra Zutat: Schafskäse, Scharf, Extra Tomate

wollen diese berücksichtigt sehen und nicht lange warten. Eine schnelle und passende Zubereitung erhöht deren Zufriedenheit, andernfalls werden sie sauer.

* Zeit + Genauigkeit wirkt sich auf Zufriedenheit aus. Je höher Zeit, desto schlechter Mood. Bestellung falsch, schlechter Mood. Bestellung richtig erhöht Mood

Die Anzahl bislang verkaufter Gerichte und die Gesamtzufriedenheit der beiden Akteurgruppen wird fortlaufen auf dargestellt.

* Zufriedenheit Kunden: Abhängig von Zeit u. Genauigkeit
* Zufriedenheit Mitarbeiter: Abhängig von Anzahl Kunden u. Arbeitszeit u. Pausen
* Verkaufte Gerichte nach Gericht sortiert

Beim Start der Simulation kann eingestellt werden wie viele Leute in der Bude arbeiten, wie viele Kundis im Durchschnitt pro Zeit kommen, wie groß die Kapazitäten von Rohmateriallager und den Behältnissen sind, wie viel Leerlauf die Mitarbeiteris brauchen, um nicht über- oder unterfordert zu sein.

* FormElement
* Uhrzeit auch darstellen?

Deine Aufgabe ist es nun, eine erste Probeversion des Döner-Trainers zu konzipieren und zu produzieren, die wenigstens das oben beschriebene sinnvoll nutzbar abbildet.

**Hinweise**

* es genügt völlig, die Bude in der Aufsicht darzustellen.
* es genügt, die Akteure als Smileys darzustellen, an deren Ausdruck die Stimmung abgelesen werden kann.
* die Zutatendarstellungen können recht abstrakt sein, solange man sie gut unterscheiden kann.
* es sollen ein oder mehrere Arrays genutzt werden, um die Personen zu verwalten.

Gehe bei der Konzeption und Produktion der Abschlussarbeit folgendermaßen vor:

**Hilfe**

* Nutze weiter Github um mit deinen Betreueris zu kommunizieren. Wir stehen bereit und freuen uns, helfen zu können. Dies gilt für die Konzeption ebenso wie für die Produktion.
* Stelle dabei sicher, dass Du die aktuellen Daten leicht zugänglich gemacht hast. Versieh die Anfrage gleich mit allen erforderlichen Links, so dass wir nicht suchen und raten müssen.
* Wenn wir Pull-Requests erstellen, 'merge' diese nicht gleich in dein Repository, sondern schaue dir die Änderungen an. Möglicherweise hinterlassen wir nur unformatierte Kommentare, die mit der Codesyntax inkompatibel sind. Sofern das mergen angebracht ist, werden wir das explizit mitteilen.

**Konzeption**

**Funktionale Analyse**

* Beschäftige dich zunächst noch nicht mit der Frage, wie Du dein Konzept umsetzen willst, sondern nur damit, welches Erlebnis der Nutzer haben soll und wie er mit der Anwendung interagiert. ANWENDUNGSFALLDIAGRAMM
* Lege die Plattform (PC, Mobil), sofern nicht vorgegeben, entsprechend nachvollziehbarer Kriterien fest und dokumentiere dies.
* Erstelle Skizzen, gerne handgezeichnet, welche darstellen, was der Nutzer während des Interaktionsverlaufs sieht. UI-SCRIBBLE
* Beschreibe, wie und welche andere Information noch zum Nutzer gelangt, beispielsweise akustisch.
* Erstelle Anwendungsfalldiagramme, für komplexere Interaktionsabläufe kannst Du auch domänenübergreifende Aktivitätsdiagramme zeichnen, mit dem Nutzer und dem/der System/e als "Schwimmbahnen" (siehe Booklet 1.2.6 Activity Partitions)
* Überlege auch, wie gegebenenfalls Objekte miteinander interagieren, und unter welchen Bedingungen bestimmte Interaktionen nicht möglich sind.

**Technische Analyse**

* Überlege nun, wie deine Anwendung umgesetzt werden könnte. Höchstwahrscheinlich wirst Du dies nicht sofort von Anfang bis Ende durchdenken und niederschreiben können, sondern wirst mehrere Iterationen durchlaufen.
* Erstelle Klassendiagramme und wende dabei die Prinzipien, Fragen und Regeln der objektorientierten Modellierung an.
* Erstelle dann Aktivitätsdiagramme auf verschiedenen Detailebenen.
* Meist empfiehlt es sich, zuerst einen groben Ablauf darzustellen, um Teilaspekte zu identifizieren. Erstelle dann für die Teilprobleme wieder Aktivitätsdiagramme. So wandert dein Fokus von „wie setzte ich die Anwendung um?“ zu „wie setze ich diesen Teil oder diesen Aspekt der Anwendung um?“.
* Im Idealfall lassen sich Probleme auf diese Art so weit aufgliedern, bis sich für alle Unterprobleme einfache Lösungen finden, und damit das Gesamtproblem gelöst ist. In allen anderen Fällen hilft dir diese Vorgehensweise zumindest, nicht über alles gleichzeitig nachdenken zu müssen und sich nicht schon am Anfang in Details zu verlieren
* Zudem kannst Du auf diese Weise relativ schnell einen Lösungsansatz strukturieren und ihn anderen kommunizieren. Gehe davon aus, dass nicht gleich der erste Versuch zum Erfolg führt, sondern dass die ersten Diagramme eher Hinführungen für weitere, verbesserte sind.

**Bedenke: die Konzeption muss derart gestaltet werden, dass andere ohne vertiefte Kenntnis der Aufgabenstellung die Anwendung auf dieser Grundlage produzieren können und mit großer Wahrscheinlichkeit das gewünschte Ergebnis erreichen.**

**Produktion**

* Erst wenn deine Konzeption ein funktionsfähiges System darstellt und obigem Merksatz nahe kommt, beginne mit der Produktion.
* Implementiere deine Abschlussarbeit syntaktisch korrekt gemäß deiner technischen Analyse und nach den festgelegten Stil-Regeln mit Hilfe von Typescript, HTML und CSS, sowie gegebenenfalls Nodejs und MongoDB.
* Teste die Applikation regelmäßig, ausgiebig und frühzeitig. Lasse auch andere Personen testen um festzustellen, ob die Anwendung bedienbar und fehlertolerant ist.
* Achte darauf, dass Du deinen Code häufig in dein Repository überträgst und dabei sinnvolle Commit-Nachrichten schreibst. Übertrage ihn dann auch gleich auf Github!

**Abgabe am 15.02.22 12 Uhr Mittags**

* Fasse die Konzeption als ein wohlformatiertes Designdokument in PDF zusammen!
* Platziere wie üblich einen Link zu Github-Pages auf die fertige und lauffähige Anwendung im Steckbrief.
* Platziere ebenso einen Link zu der Stelle in deinem Github-Repository, an dem der Quellcode und das Designdokument zu finden ist.
* Stelle zudem auf diese Art auch ein gepacktes Archiv zur Verfügung, welches folgende Daten enthält
  + Das Designdokument
  + Die Projektordner inklusive aller erforderlichen Dateien, also auch Bild- und Audiodaten
  + Eine kurze Anleitung zur Installation der Anwendung unter Berücksichtigung der erforderlichen Dienste (z.B. Heroku, MongoDB etc.)
  + Eine kurze Anleitung zur Interaktion mit der Anwendung

Customer und Employee als Klasse

**Employee**

Was hat es? (Eigenschaften und Attribute)

* Appereance (3 verschiedene physische Erscheinungen)
* Mood (5: konzentriert, gestresst, müde, unterfordert, glücklich) mit eNum arbeiten?
* Position: Vector
* Size: number
* breakNeeded
* working

Was kann es? (Methoden)

* Draw
* changeMood
* doBreak
* orderIngredient
* prepareIngredient
* changeWorkWithEmployee (?)
* move (Auf abstand zu anderen Mitarbeitern achten, im Küchenbereich aufhalten)

Was kann es in Wechselwirkung mit den Kunden? (Methoden)

* Bestellung entgegenehmen (takeOrder)
  + Zusätzlich informieren wenn aus ist
* makeFood
  + Bestellung zusammenstellen
  + Bestellung abschließen und Kunden geben (Hauptprogramm überprüft Richtigkeit)
* Kunden besänftigen ?
* Kunden anschreien ?

Was weiß es?

* Spezielle Parameter

Was ist es?

Was kümmert sich?

**Customer**

Was hat es? (Eigenschaften und Attribute)

* Appereance (5 verschiedene physische Erscheinungen)
* Mood
* Position: Vector
* Size
* Patience
* Order

Was kann es? (Methoden)

* Draw
* changeMood
* move
* wait
* Bestellung aufgeben (placeOrder)

**Vector (Klasse)**

Wie gehabt

**Humanbeing (Superklasse)**

Vielleicht als Superklasse von Customer und Employee

Container als bewegbare Objekte

**Bestellung**

1. Kalb/Hähnchen/Falafel +
2. Döner/Dürüm +
3. Zutaten: mit allem (beinhaltet: Salat, Tomate, Zwiebeln, Kraut, Peperoni, Mais, Soße):, oder ohne Zutat XY
4. Extrawunsch (optional): Extra Scharf, Mit Schafskäse, Extra Zutat XY

* 4 Arrays die zusammengeführt werden siehe aufgabe

**Sonstige Gedanken**

Bestellung ein random generierter Zusammenschluss aus mehreren Array

Employee und Customer in Array im Hauptprogramm speichern

Wo wird Lager gespeichert?

10,5+10+11,5 =32