

# Endabgabe - DönerDream

# Konzeptdokument

Modul: Entwicklung Interaktiver Anwendungen II

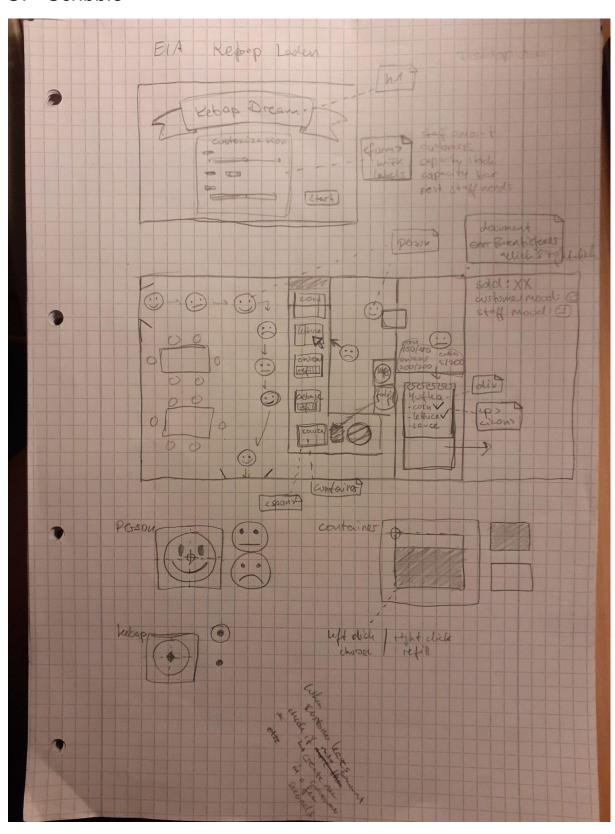
Studiengang: MKB 2 / Gruppe B Matrikelnummer: 268339

vorgelegt von

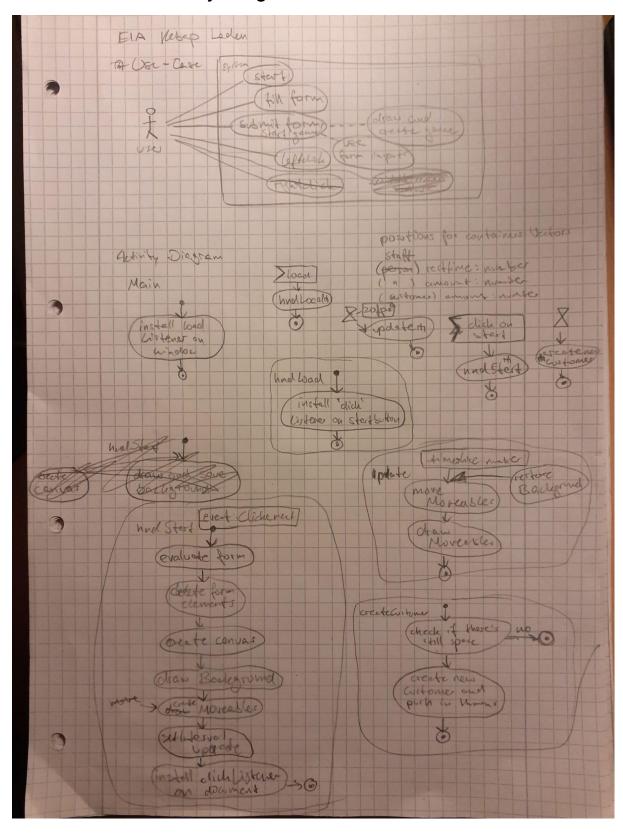
Amélie Dell'Oro

# **Erste Schritte**

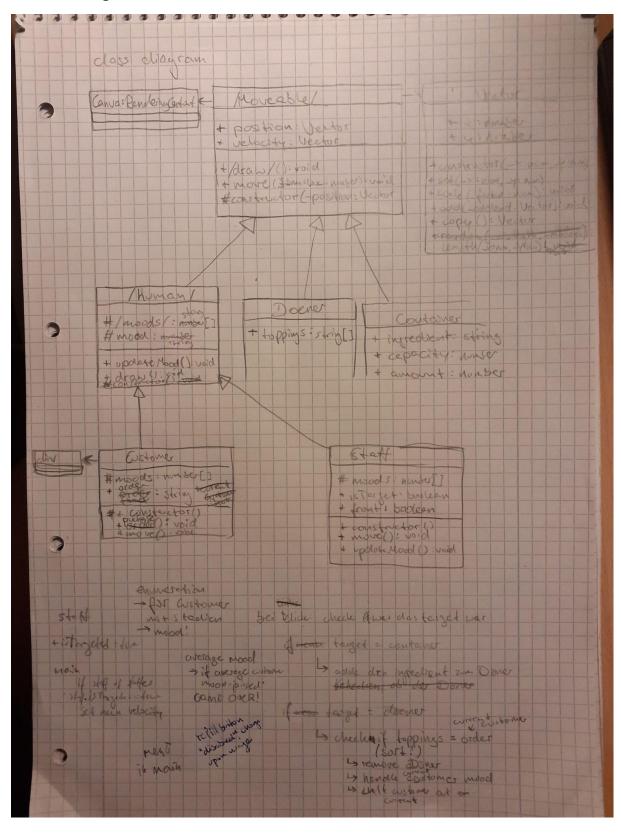
### UI - Scribble



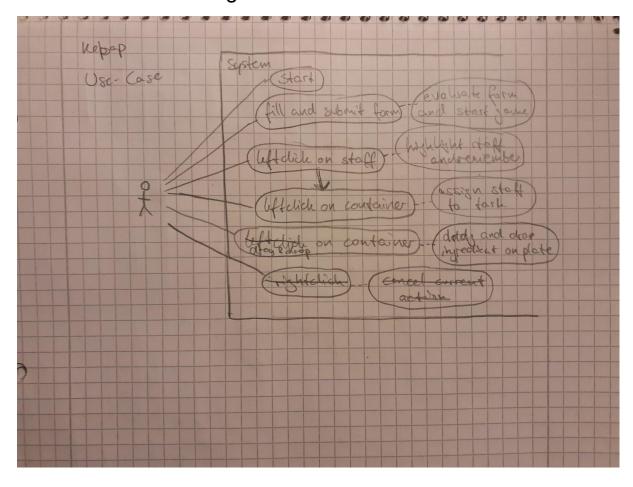
#### Use Case und Activity Diagram



#### Class Diagram



### Zweites Use Case Diagram



Class Diagram und Activity Diagram Class Diagram CanvasRenderingContext Vector Moveable + position: Vector + x: number # velocity: vector + y: number # constructor(\_position: Vector) + constructor(\_x: number, \_y: number) + move(\_timeslice: number): void + set(\_x: number, \_y: number): void + draw(): void + scale(\_factor: number): void + add( addend: Vector): void + copy(): Vector + get length(): number Container Human **Plate** + ingredient: string # moods: string[] + contents: string[] # mood: string + capacity: number + amount: number + constructor() # constructor(\_position: Vector) + draw(): void + constructor(\_position: Vector) + draw(): void + draw(): void # updateMood(\_addend: number): void Staff <<TASK>> <<STATE>> Customer # moods: string[] Bar GoingToRefill + task: TASK # moods: string[] In queue + target: Container | undefined Refilling + order: string + state: STATE waiting ComingFromRefill + active: boolean; leaving + restingTime: number; Returning Waiting - originalPosition: Vector; + constructor() Recovering + move(\_timeslice): void + constructor() + receiveFood() + move(\_timeslice): void + updateMood(): void + updateMood(\_addend: number): void + refill(\_target: Container): void + fillPlate( container: Container): void Activity Diagram - Moveable timeslice: number position: Vector constructor add velocity \* timeslice set position to to position position set velocity to 0,0 Activity Diagram - Human position: Vector addend: number constructor updateMood if \_addend negative, super(\_position) check that index of draw Smiley this.mood > 0representing mood change this.mood by addend Activity Diagram - Customer \_position: Vector timeslice: number constructor move every 10s super(\_position) super.move(\_timeslice) set velocity, mood switch this.state and state updateMood if position outside of [leaving] {in queue} if position.x above half the pick ingredients and canvas dimension canvas, set velocity(0,0, set splice this out of state = waiting and printOrder() set order customer array setInterval for else if there's someone infront updateMood() if they're too close, set velocity(0,0) else keep walking \_plate: string[] receiveFood stop updateMood Interval updateMood evaluate plate according to this.order switch this.mood [pissed] super.updateMood set velocity to down and state = leaving according to overlap set velocity to down super.updateMood(-1) and state = leaving Activity Diagram - Staff timeslice: number constructor move super.move(\_timeslice) super(randomPosition within reason) switch this.state else if close enough to mood = content and if close enough to target kitchen, set velocity(0,0) task = waiting set velocity towards and task = refilling and [ComingFromRefil] √goingToRefill/ originalPosition, task = if close enough to target, set setTimeout for Returning and target = velocity towards kitchen this.target.amount = setInterval for undefined Returning] capacity and task = updateMood(1) comingFromRefill if close enough to original position, set velocity(0,0) and state = waiting stop Interval for updateMood and setInterval for updateMood(1) addend: number container: Container target: Container updateMood fillPlate refill switch this.mood push \_container.ingredient set task = refill, set target = \_target, into plate.contents set velocity towards target.position remember current task, set task = [burnout] substract 1 of Recovering, set velocity(0,0) and stop Interval for updateMood and container.ingredient after restingTime resume task setInterval for updateMood(-1) if time between super.updateMood(\_addend) actions too small, updateMood(-1) Activity Diagram - Main animationframe formData: formData; click load customerSpawnrate: number; on startbutton staffRestingTime: number; staffAmount: number; install ClickListener install LoadListener prepareGame maxStock: number; on startButton on window containerCapacity: number; barPosition: Vector; middle: Vector; kitchen: Vector; background: ImageData customer crc2: CanvasRenderingContext customers: Customer[] = [] click click SpawnRate workers: Staff[] = [] on restockButtons prepareGame on canvas foods: Container[] = [all containers] plate: Plate = new Plate newCustomer interface for Stock; hndCanvasClick restock 📥 find and evaluate form stock: Stock; and set variables to stocks: HTMLDivElement; according parameters order: HTMLDivElement; clear HTML body event: ClickEvent startGame update restock create canvas element target button add attribute "disabled" push staffAmount x restore background get crc2 new Staff(restingTime) in workers with first after 5s worker state = bar move(frameTime set middle.x and 1000) and draw() all middle.y to half the Moveables canvas dimensions requestanimationframe for update set targeted stock to maxStock draw background according to UI scribble setInterval for newCustomer by and save as background customerSpawnRate install ClickListener on canvas newCustomer if amount of customers < 5 push new Customer into create stocks div and customers display stock / maxStock create buttons according to event: ClickEvent stock and addAttribute hndCanvasClick "disabled" get mouse position add ClickListener relative to canvas element [inside container that √on waiting staff append buttons to div [a staff is active] staff.active = is no staffs target and div to canvas activeStaff.refill(container) true on plate] create stats div and barStaff.fillPlate(container) display game stats append div to body if a staff is active, set active=false create order div and else do nothing append to canvas find waiting customer and empty plate contents receiveFood(plate.contents)