

# **ПРОЕКТ ПО ПРЕДМЕТОТ ЕЛЕКТРОНСКА И МОБИЛНА ТРГОВИЈА**

Веб апликација за онлјан нарачка на храна

Изворен код достапен [тука](#)

Изработено од:

Симона Стојановска 181036

Септември, 2021

## Вовед

„My Restaurant“ е веб апликација која овозможува преглед и нарачка на јадења од мени на некој ресторан. Апликацијата овозможува регистрација и најава на корисници. Корисниците може да го прегледуваат менито. Освен што ги прегледува јадењата корисникот може да одбере некое јадење и истото да го додаде во својата нарача. Откако ќе заврши се селектирањето јадења кои сака да ги порача, корисникот плаќа за својата нарачка. Веб апликацијата овозможува преглед на сите нарачки кои се подготвени за достава. Во продолжените се прикажани слики кои соодветствуваат на секое прашање од Лаб 03, а под секоја слика е напишано дали има некои измени при имплементација или не.

Customer микросервисот работи на порта 8082

Delivery микросервисот работи на порта 8083


Menu микросервисот работи на порта 8084

Order микросервисот работи на порта 8085

Редоследот на извршување на микросервисите е Customer, Menu, Order, Delivery

## Сценарио кое се обработува

Опишете го сценариото кое го обработувате (200 збора).



Сценариото кое ќе го обработувам е систем за менаџирање на некој ресторан. Конкретно ќе се осврнам на делот за креирање на нарачки и нивна достава. Корисниците ќе го користат системот online. Тие би можеле да го разгледуваат менито и да селектираат некое јадење од него. За секое јадење се чува информација за тоа од кои состојки е приготвено. Откако корисникот ќе селектира некое јадење истото се додава во неговата корисничка кошничка. На овој начин се креира нова нарачка. Во една нарачка може да се додадат повеќе јадења. Откако ќе заврши со нарачката, корисникот ја внесува адресата на која сака да се достави нарачката. Информациите за нарачката и адресата за достава се предаваат на одделот за достава. Значи системот овозможува управување со ресторанот, конкретно со јадењата, со нарачките, со корисниците и со доставата на нарачките.

## ЕА дијаграм

Идентификувајте ги ентитетите и релациите меѓу нив кои ви се потребни за имплементација на сценариото. Бројот на дефинирани ентитети мора да биде најмалку 3. Нацртајте го и ЕА дијаграмот и прикачете го како слика.

↶ A ▾ B I ≡ ≡ 🔑 🔄 🖼️

Ентитетите кои ги имам дефинирано се: Customer, Order, OrderItem, Meal, Ingredient, Delivery. Како што кажува и самото име на ентитетот, Customer ги означува корисниците, Order е нарачката, OrderItem е ставка во нарачката, Meal го репрезентира јадењето, Ingredient состојките од кои е направено, а Delivery ентитетот се однесува на доставата, односно на информациите поврзани со доставата.

Релациите помеѓу ентитетите се: Customer креира Order. Еден клиент може да креира N нарачки, но една нарачка му припаѓа само на еден клиент.

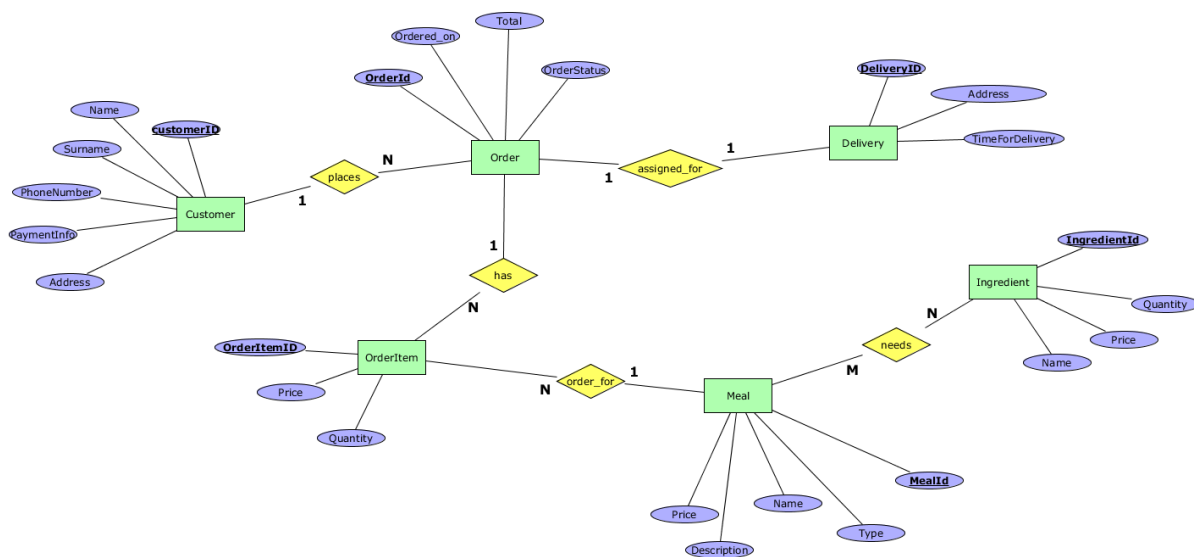
Секоја нарачка се состои од ставки (OrderItem). Една нарачка може да има N ставки, но една ставка припаѓа само на една нарачка.

Секоја ставка се состои од јадење (Meal). Едно јадење може да се јави во N ставки, но во една ставка има само едно јадење.

Секое јадење е зготвено со состојки (Ingredient). Во едно јадење може да има N состојки и една состојка може да се користи во M јадења.

Секоја нарачка се назначува за достава (Delivery). Една нарачка може да се достави само еднаш. Една достава се однесува само на една нарачка (бидејќи во Delivery ентитетот се чува адреса на која треба да се достави нарачката и време на достава).

Ентитетите, нивните атрибути и релациите помеѓу ентитетите се прикажани на ER дијаграмот кој е прикачен подолу.



Направена е мала измена во ЕА дијаграмот. Конкретно ентитетот Состојки,Ingredients е моделиран како вредносен објект, наместо како ентитет. При дефинирање на ограничените контексти, дефинирав еден ограничен контекст кој се однесува на самиот ресторан. Во тој ограничен контекст беа ентитетите Meal и Ingredient, а како Aggregate Root го одбрав Meal агрегатот. Со тоа сите операции поврзани со состојките ќе треба да поминуваат преку овој агрегат. Значи секогаш кога додаваме некој состојка, треба да дефинираме јадење од кое е дел, а потоа информации за самата состојка. За да го одбегнам тој проблем имам две можности, да дефинирам посебен ограничен контекст за состојките или да ги моделирам како вредносен објект за секој јадење.

Бидејќи апликацијата се фокусира на нарачка на храна, се одлучив состојките да ги моделирам како вредносен објект.

## Атрибути на ентитети

Идентификувајте ги потребните атрибути за секој од ентитетите. За секој од атрибутите дефинирајте го типот.

↴

A ▾

B

I

≡

≡

🔑

🔗

🖼️

Ентитетите и атрибутите кои ги имам дефинирано се прикажани во ER дијаграмот.

Еден од ентитетите е Customer. Тој ги има атрибутите CustomerId (id на ентитетот од тип long, кој се зема за примарен клуч), Name (име на корисникот, од тип String), Surname (презиме на корисникот, од тип String), PhoneNumber (телефонски на корисникот, од тип String), PaymentInfo (кредитна картичка на корисникот, од тип String), Address (адреса на живеење на корисникот, од тип String).

Друг ентитет е Order. Неговите атрибути се OrderId (id на ентитетот од тип long, кој се зема за примарен клуч), Ordered\_on (датум на кој е креирана нарачката, од тип date), Total (вкупната сума на нарачката), Order\_status (статус на нарачката, енумерација).

Друг ентитет е OrderItem со атрибути OrderItemId (id на ентитетот од тип long, кој се зема за примарен клуч), Price (ја означува цената на ставката од тип double), Quantity (ја означува количината на ставката, од тип int).

Следен ентитет е Meal. Тој ги има атрибутите MealId (id на ентитетот од тип long, кој се зема за примарен клуч), Name (име на јадењето од тип String), Description (опис на јадењето од тип String), Price (цена на јадењето од тип double), Type(тип на јадење, се чува како енумерација).

Следно имаме ентитет Ingredient со атрибути: IngredientId (id на ентитетот од тип long, кој се зема за примарен клуч), Name (име на состојката, од тип String), Price (ја означува цената на состојката од тип double), Quantity (ја означува количината на состојката во магацин, од тип int).

И на крај имаме ентитет Delivery DeliveryId (id на ентитетот од тип long, кој се зема за примарен клуч), Address (адресата на која треба да се достави нарачката од тип String), TimeForDelivery (време на достава на нарачката, од тип date).

## Ограничени контексти

Идентификувајте ги ограничените контексти (bounded contexts) во вашето сценарио.

↴

A ▾

B

I

≡

≡

🔑

🔗


🖼️

За системот за нарачка на храна во еден ресторан имаме неколку ограничени контексти. Првиот ограничен контекст се однесува на менаџирање со нарачки. Во овој контекст се ентитетите Order и OrderItem. Друг ограничен контекст се однесува на самиот ресторан. Во него имаме 2 ентитети, тоа се Meal и Ingredient. Следниот ограничен контекст се однесува на доставата. Во него го имаме само ентитетот Delivery. И последниот ограничен контекст се однесува на корисниците. Тука го имаме само ентитетот Customer.

Како што напишав и погоре, во ограничениот контекст кој се однесува на ресторанот имаме само еден ентитет, Meal. Соодветно истиот се зема на Aggregate root, како што беше и дефинирано.

## Агрегати


Идентификувајте ги агрегатите во секој од ограничените контексти.



Претходно дефинирам 4 ограничени контексти. Едниот се однесуваше на нарачки, другиот на ресторанот, третиот на корисниците и четвртиот на достава. Во ограничениот контекст на нарачки имаме 2 ентитета, Order и OrderItem. Тие два ентитета претставуваат агрегати. Во контекстот кој се однесува на ресторанот имаме 2 ентитета Meal и Ingredient кои исто така претставуваат агрегати. Во ограничениот контекст за корисници имаме еден ентитет, Customer, кој претставува агрегат. Слично и во ограничениот контекст за достава имаме само еден ентитет Delivery, кој претставува агрегат.

## Aggregate root

Идентификувајте го Aggregate Root на секој од идентификуваните агрегати.



Претходно дефинирам 4 ограничени контекст, за нарачки, за ресторан, за корисници и за достава. Во ограничениот контекст на нарачки имаме 2 агрегати, Order и OrderItem. Како aggregate root се зема Order агрегатот и за целата комуникација со контекстот за управување со нарачки ќе се користи Order агрегатот. Во контекстот кој се однесува на ресторанот имаме 2 агрегати Meal и Ingredient. Како aggregate root се зема Meal, бидејќи тој ги претставува јадењата кои корисниците може да ги нарачаат. Па така секогаш кога додаваме некое јадење во нарачката имаме комуникација помеѓу aggregate root на контекстот на нарачки и на ресторанот. Во ограничениот контекст за корисници имаме еден агрегат, Customer, кој претставува aggregate root. На крај во ограничениот контекст за достава имаме еден агрегат, Delivery кој се зема за aggregate root.

## Бизнис правила

Идентификувајте неколку правила за конзистентност (бизнис правила) во сценариото. Специфицирајте кој ентитет ќе ги поседува имплементациите на истите?

↕

A ▾

B

I

≡

≡

🔗

🔄

🖼️

Customer ентитетот ќе има имплементација на регистрација и најава на корисници.

Menu ентитетот ќе овозможи преглед на достапните јадења, додавање, бришење, менување на јадење. Дополнително може да имаме листа на јадења за кои ќе биде дефиниран попуст. Корисникот ќе може да селектира јадење и количина, со што се креира нарачка.

Order ентитетот ќе овозможи креирање на нова нарачка, додавање на ставки (OrderItem) во нарачката. Откако ќе заврши со нарачката, истата се затвара (OrderStatus=closed). По ова корисникот може да ја внесе адресата на која треба да се изврши доставата. Информациите за нарачката и адресата на достава се користат за креирање на Delivery објект.

Meal ентитетот овозможува преглед на достапните јадења и додавање на ново јадење во менито. Исто така, корисникот селектира јадење и количина и тоа се додава како ставка во нарачката.

## Вредносни објекти

Идентификувајте неколку вредносни објекти (value-objects) во вашето сценарио. Кои методи би ги имплементирале?

↕

A ▾

B

I

≡

≡

🔗

🔄

🖼️

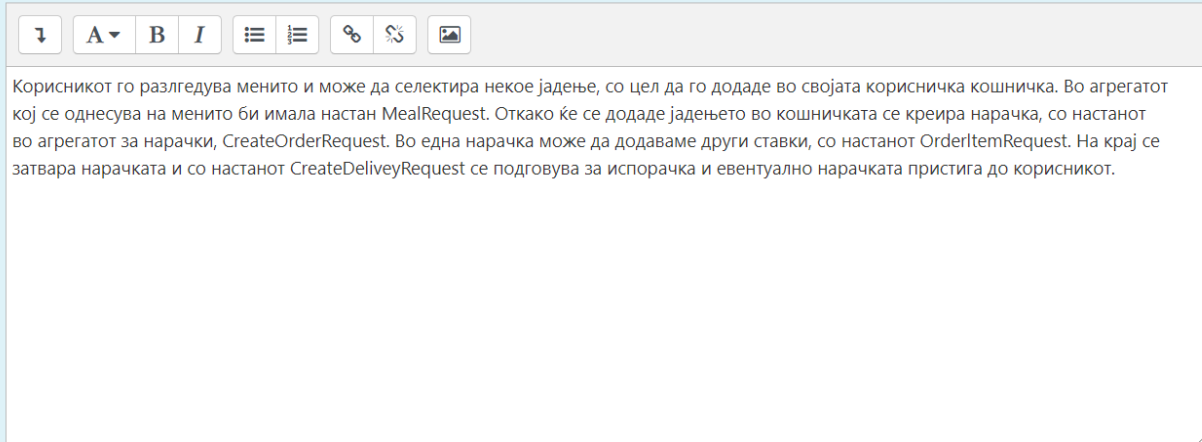
Во ова сценарио имаме неколку вредносни објекти. Едниот од нив, ќе го именувам како Money. Овој вредносен објект би го имале ентитети Order (атрибутот Total), OrderItem (атрибутот Price), Meal (атрибутот Price), Ingredient (атрибутот Price). Ќе ги имплементираат основните операции за работа со пари, додавање, одземање, множење.

Другиот вредносен објект кој може да го имплементирам е PaymentInfo за еден корисник. Него би го имал ентитетот Customer (атрибутот PaymentInfo). Би се чувале број на кредитна картичка, име и презиме на корисник на картичка, датум на важење на картичката и кодот. Генереално би имале проверка на валидност на внесено податоци и одземање на средства од картичката при успешна наплата на нарачката.

Следниот вредносен објект е Address. Овој вредносен објект би го имале ентитети Customer (атрибутот Address), Delivery(атрибутот Address). Во овој вредносен објект би чувале улица, број, град, држава.

## Настани

Идентификувајте неколку настани (events) кои треба да протекуваат помеѓу агрегатите.



The screenshot shows a rich text editor with a toolbar at the top containing icons for undo, font color, bold, italic, bulleted list, numbered list, link, unlink, and image. Below the toolbar is a text area containing the following text:

Корисникот го разгледува менито и може да селектира некое јадење, со цел да го додаде во својата корисничка кошничка. Во агрегатот кој се однесува на менито би имала настан MealRequest. Откако ќе се додаде јадењето во кошничката се креира нарачка, со настанот во агрегатот за нарачки, CreateOrderRequest. Во една нарачка може да додаваме други ставки, со настанот OrderItemRequest. На крај се затвара нарачката и со настанот CreateDeliveyRequest се подготвува за испорачка и евентуално нарачката пристига до корисникот.

Кога корисникот ќе додаде некое јадење во својата нарачка се публикува настанот MealAddedInOrder, со што бројот на нарачки за соодветното јадење се зголемува. Инаку за секое јадење се дефинира број на нарачки кој би можел да го користат вработените за приготвување на нарачката. Се секое додавање на ставка во нарачката овој број се зголемува. Доколку корисникот избрише некоја ставка од нарачката се публикува настанот MealRemovedFromOrder, со што бројот на нарачки за јадењето се намалува. Овие два настани ги публикува Order микросервисот кој Meal микросервисот. Откако корисникот ќе ја плати нарачката се публикува настанот SuccessfullyPaidForOrder со што треба да се креира Delivery објект. Овој настан го публикува Order микросервисот кон Delivery микросервисот. Откако нарачката ќе биде доставена, ќе се заврши со достава, се публикува настанот OrderSuccessfullyDelivered со што се менува статусот на нарачката. Овој настан го публикува Delivery микросервисот, кој Order микросервисот.