Електронска и мобилна трговија 2020 – Изработка на проект за лаб. вежби – Фаза 1	
Индекс, име и презиме на членовите од тимот	1. 161555, Антон Кахуаџи 2.
Наслов на проектот	ЈСП Билети

Дефинирајте сценарио треба да го имплементирате со користење на алатките од Domain Driven Design и кое треба да биде по сложеност слично на сценариото на аудиториски вежби. Како потсетување, користете ги презентациите за Strategic Design и Tactical Design кои се поставени на курсот. Во фаза 2 истото сценарио ќе треба да го испрограмирате.

1. Опишете го сценариото кое го обработувате (200 збора).

   Ако ЈСП (Јавното сообраќајно претпријатие Скопје) реши да се прошири во бизнисот на возови и трамваи ќе можи да се искористи оваа идеа. Ќе имаме билети кој ќе ги купуваат луѓето, тие билети ќе можат да бидат искористени повеќе пати. Ќе имаме TrainRides, тие ќе се неделни патувања кој се во одредени денови, времиња, и под одредени рути. Ќе имаме контролори кој на почеток од едно возење, пред да се тргни или при тргнување, ги чекираат тие билети, со тоа се додава тој TrainRide во листата на потврдени CompletedRides, и во исто време се земаат тие билети и на секој од нив намалува бројот на искористеност за 1. Ако не е важечки некој билет, а тоа е, ако нема доволно средства за да се чекира, тогаш нема да успее чекирањето на тој Ride, и со тоа ќе мора контролерот да се обиди од ново, со тоа што ќе дознае кој билет не е важечки за да се извести тој патник.

2. Идентификувајте ги ентитетите и релациите меѓу нив кои ви се потребни за имплементација на сценариото. Бројот на дефинирани ентитети мора да биде најмалку 3.

   Ќе ги имаме следните ентитети:

   a. Conductor: Основни податоци за еден контролер.
   b. Ticket: Информации за билет, кој Cusomter, го купил, искористувања.
   c. Ride: Рута Route, време и денови кога ја има, станици Stops.
   d. CompletedRide: Информации за кога завршил еден Ride, кој контролер го чекирал, и кога.
   e. Stop: Име, број, и локација.

3. Идентификувајте ги потребните атрибути за секој од ентитетите.

   Ќе ги чувам следните атрибути:

   a. Conductor:
      i. Id: int
      ii. Name: string
      iii. Employed_Since: LocalDate
   b. Ticket:
      i. Id: int
      ii. Customer: <Embedded> Customer
      iii. Bought_On: LocalDateTime
      iv. Expiry_Date: LocalDate

        v. Usages: <embedded> Usages
    c. Ride:
        i. Id: int
        ii. Route: <Embedded> Route
        iii. Occurrence: List<TimePoint>
        iv. Stops: List<Stop>
    d. CompletedRide:
        i. Id: int
        ii. Ride: Ride
        iii. Completed_On: LocalDateTime
        iv. Conductor: Conductor
    e. Stop:
        i. Id: int
        ii. Name: string
        iii. Number: string
        iv. Location: <Embedded> MapPoint

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

   Ќе имам еден модул кој ќе се грижи за издавање билети.

   Ќе имам друг модул кој преку него ќе се проверуваат сите Rides. Тука ќе има event за чекирање на билети и Rides.

   Ќе имам трет модул за менаџирање на рути, станици, и креирање на Rides.

5. Идентификувајте ги агрегатите во секој од ограничените контексти.

   Во првиот модул, за билети, немам комплексни агрегати, само билети сами по себе, додека информациите за патникот се embedded, и исто така за искористувањата (почетни и преостанати) се embedded.

   Во вториот модул. CompletedRide е агрегат, кој содржи Ride под-агрегат, и контролерот. Но тука имаме RideCheck агрегат, кој се испраќа како event, тој содржи Ride под-агрегат, контролерот, време на чекирање, и листа од билети.

   Во третиот модул, еден Ride е агрегат кој содржи внатре станици, листа од временски точки во неделата кога вози, и embedded податоци за рута. Исто така можат и станиците да се бројат како агрегат зошто се листа од станици.

6. Идентификувајте го Aggregate Root на секој од идентификуваните агрегати.

   За агрегатот билет, сам по себе си е Aggregate Root.

   За CompletedRide, тој е Aggregate Root врз Ride и контролер.

   За RideCheck, тука е Ride Aggregate Root.

   За агрегатот Ride, тој е Aggregate Root брз станиците, додека една станица си е сама по себе и Aggregate Root.

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

   CompletedRide треба да провери дали сите билети се важечки, така што можат да се искористат. Треба да провери дали се извршува чекирањето во времето на тој Ride.

   За секој билет треба да се провери дека искористувањата се соодветни со почетните, и дека датумот на истекување не е поминат.

   За една станица треба да се провери дека локацијата е валидна координата.

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

   Следните value-objects ги имам:

   - Customer:
     - Name: string
   - Usages:
     - Initial: int
     - Left: int
     - canTakeRide() -> bool { Left >= 1 }
     - takeRide() -> void { decrement Left }
   - TimePoint:
     - Time: Time
     - Day: DayOfWeek
   - Time:
     - Hours: int
     - Minutes: int
   - DayOfWeek:
     - Enumeration: Monday, Tuesday, …
   - Route:
     - Name: string
     - Number: string
     - Length: Distance
   - Distance:
     - Meters: int
   - MapPoint:
     - Latitude: double
     - Longitude: double
     - distanceTo(other: MapPoint) -> Distance { calculate_distance(..) }

9. Кои се предностите при користење на вредносни објекти (value-objects)?

   Подобро мапирање спрема потребата на бизнисот, подобро преставување, и поголема флексибилност. Исто така дозволува да се прошират со методи и функции по потреба. Подобра репрезентација, на пример, во Route чуваме length како тип Distance, кој во себе чува метри. Length не се грижи за како е пресметан или како внатрешно се чува.

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

    Чекирањето на билети за да тргни еден воз се прави преку event. Исто така издавањето на билети.

    На пример за чекирање на билети, прво се пуштаат events за секој билет да се искористи еднаш, па потоа да се запиши тој Ride како CompletedRide.