MovieTheaterApp

https://github.com/bunlusarac/MovieTheaterApp/

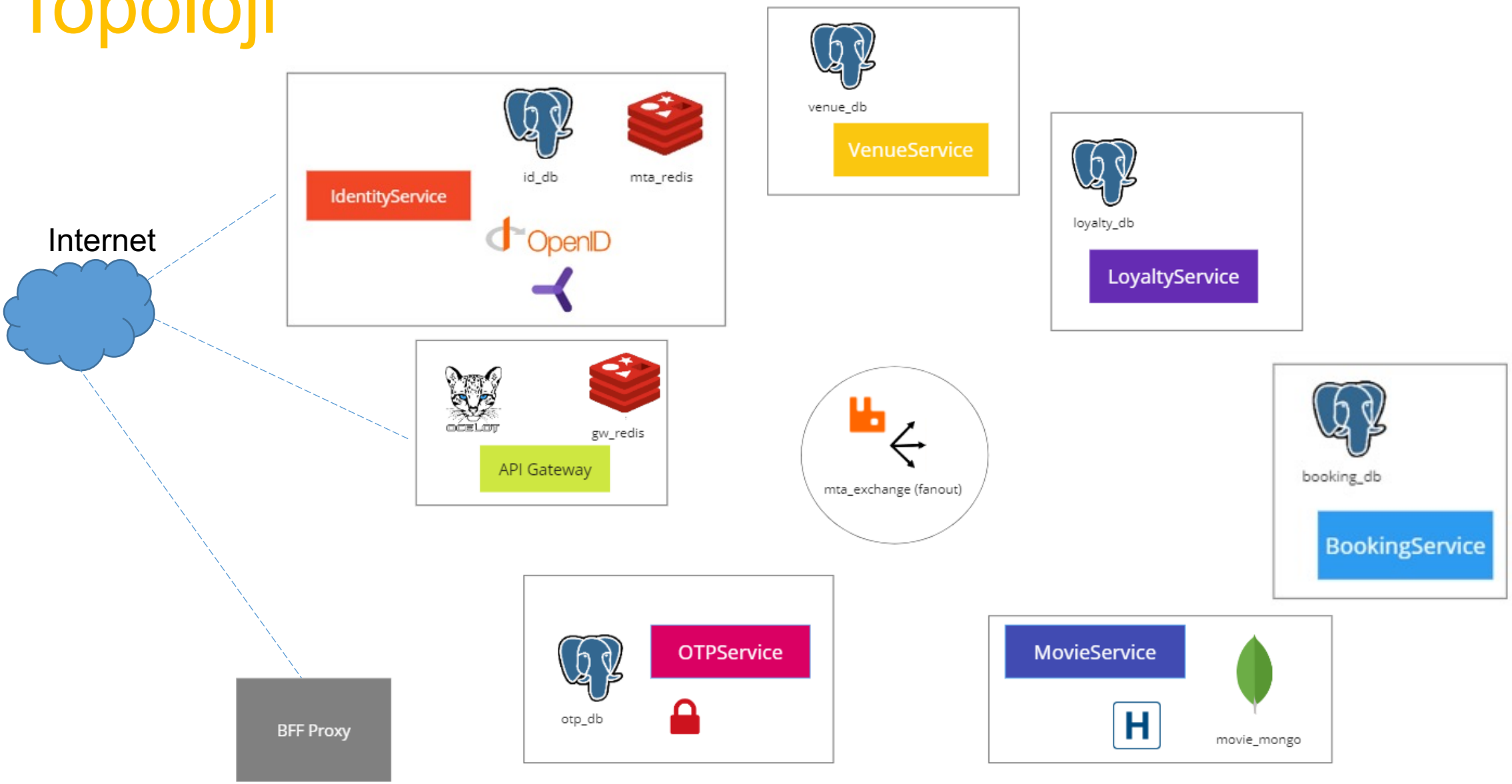
MovieTheaterApp

- Kullanıcılar istedikleri sinema mekanındaki film seanslarına bilet alabilir.
- İndirim kampanyalarından sadakat puanlarını harcayarak yararlanabilirler ve bilet alınca puan kazanabilirler.

Amaç

- Karmaşık business logic ve DDD modelleme
- Mikroservis mimarisini deneyimlemek
- Concurrency kontrolünü deneyimlemek
- RabbitMQ, Redis ve Mongo’yu kullanmak
- AuthN ve AuthZ öğrenmek ve bilgi edinmek
- CQRS kullanmak

Topoloji

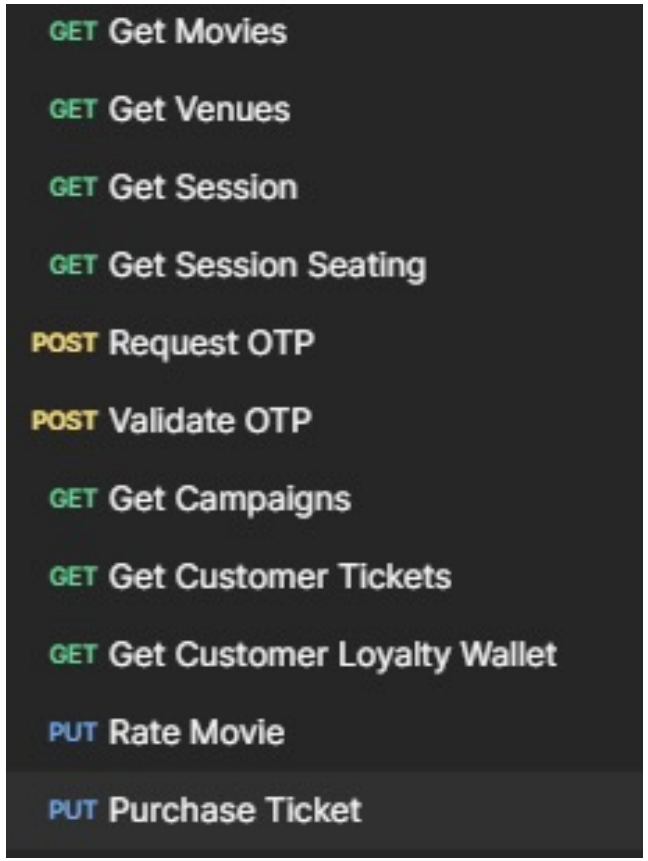

Internet
IdentityService
id_db
mta_redis
OpenID
venue_db
VenueService
loyalty_db
LoyaltyService
gw_redis
API Gateway
mta_exchange (fanout)
booking_db
BookingService
BFF Proxy
otp_db
OTPService
MovieService
movie_mongo

API Gateway

- Sadece client’ın kullanacağı endpointleri dışarıya expose eder
- Belirli endpointlerde short (DelegatingHandler) ve long session authenticationlarını (access token ile) sağlar.
- Belirli endpointlerde user-bazlı rate limiting yapılmasını sağlar. Bunun için bir Redis store’u vardır.

Redis’teki Rate limiting entrysi

Gatewayden expose edilen endpointler



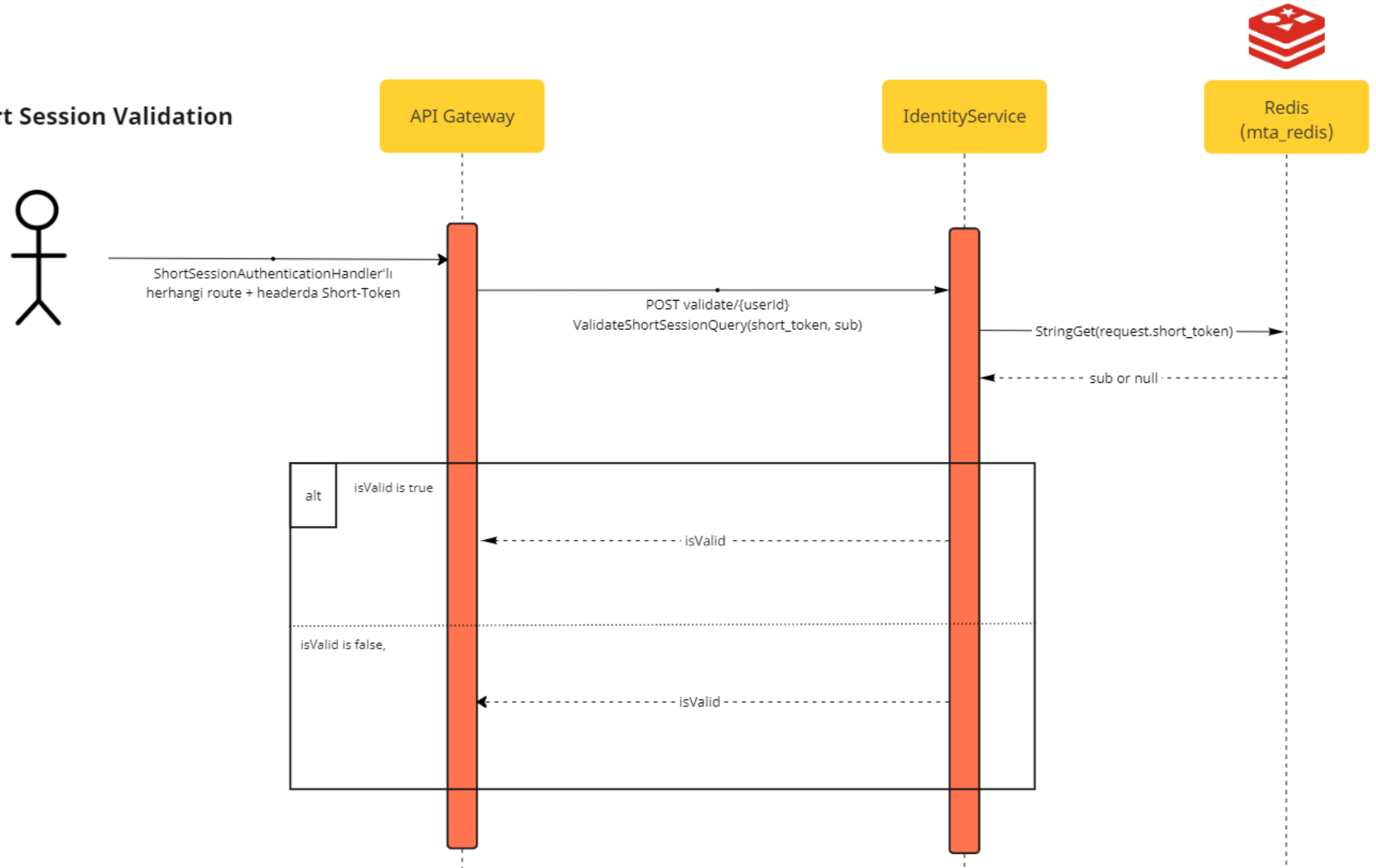
- GET /movie
- GET /venue
- GET /venue/{venueId}/session
- GET /venue/{venueId}/theater/{theaterId}/session/{sessionId}/seating
- POST /otp/request (Bearer lazım, long session)
- POST /otp/validate (Bearer lazım, long session)
- GET /campaign
- GET /ticket (Bearer lazım, long session)
- GET /customer/{customerId}/wallet (Bearer lazım, long session)
- PUT /movie/{movieId}/rate/{rating} (Bearer lazım, long session)
- PUT /ticket/purchase (Bearer ve Short-Token lazım, long ve short)

IdentityService

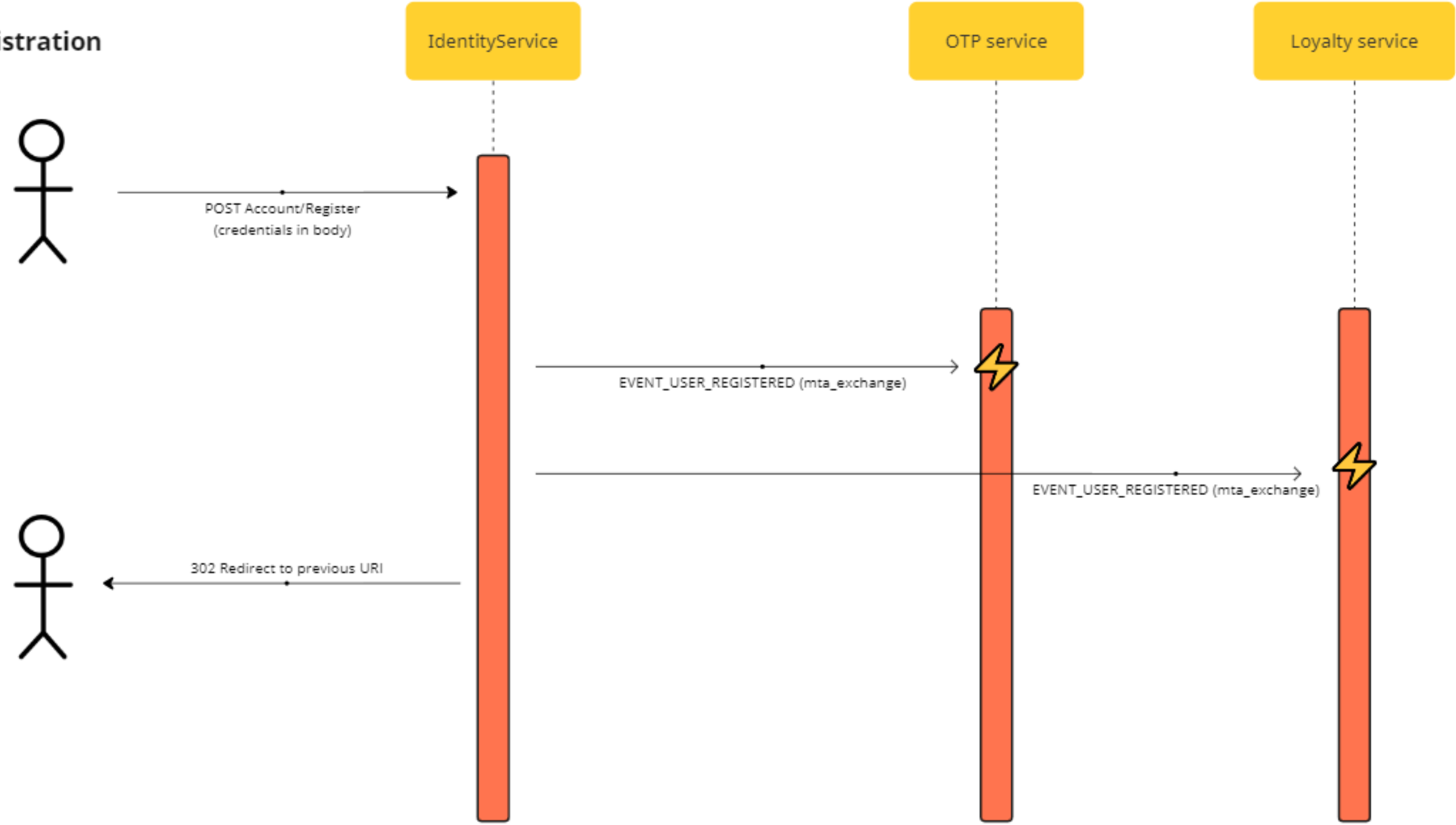


- IdentityService, Authorization Code + PKCE flowu ile authentication sağlayan OpenID Connect identity providerıdır
- Ek olarak başarılı OTP sonucu oluşan short sessionları yönetir
- Gateway'in short session token'ın geçerliliğini sorgulamasını sağlar. POST /short-session/validate/{sub}
- Kullanıcılar ASP.NET Core Identity modeli ile DB'de saklanır. ApplicationUser öğrencilik durumu ve doğum tarihi bilgisi için extend ettim.
- Registration, Login, Logout sayfaları Razor Page scaffoldinglerini kullandim ve GSM, DOB claimleri için modifiye ettim.

Short Session Validation

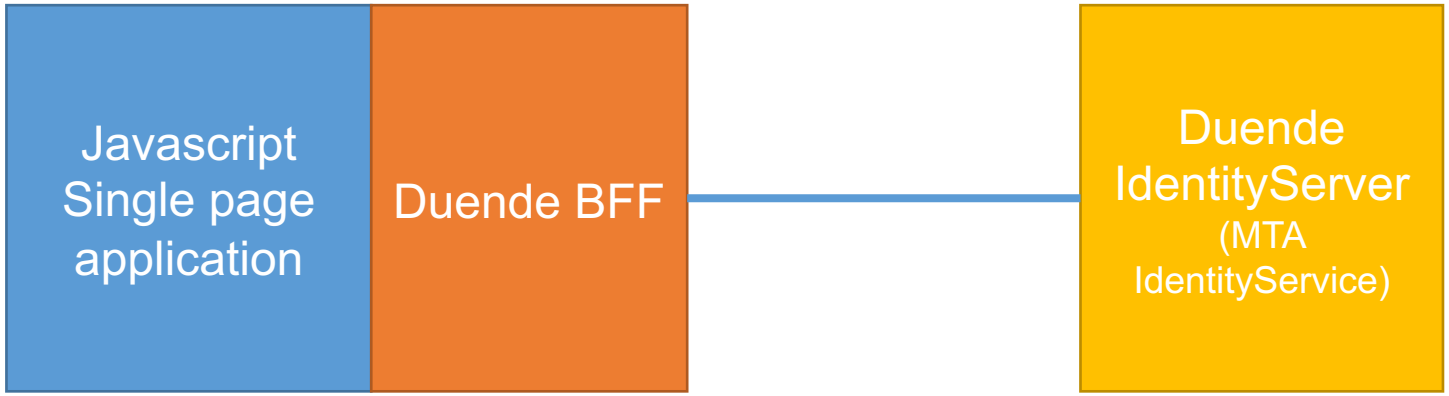


Registration



Client ve AuthN

- Client Javascript SPA'i, Duende BFF Proxy'si tarafından statik olarak hostlanır.
- Duende BFF, IdentityServer ile SPA arasında iletişim sağlar ve token saklamasını/yönetimini clienttan proxy'e taşır.
- SPA'ın OpenID Connect endpointlerini kullanabilmesi için endpointler sağlar. bff/login, bff/user gibi.



MovieService

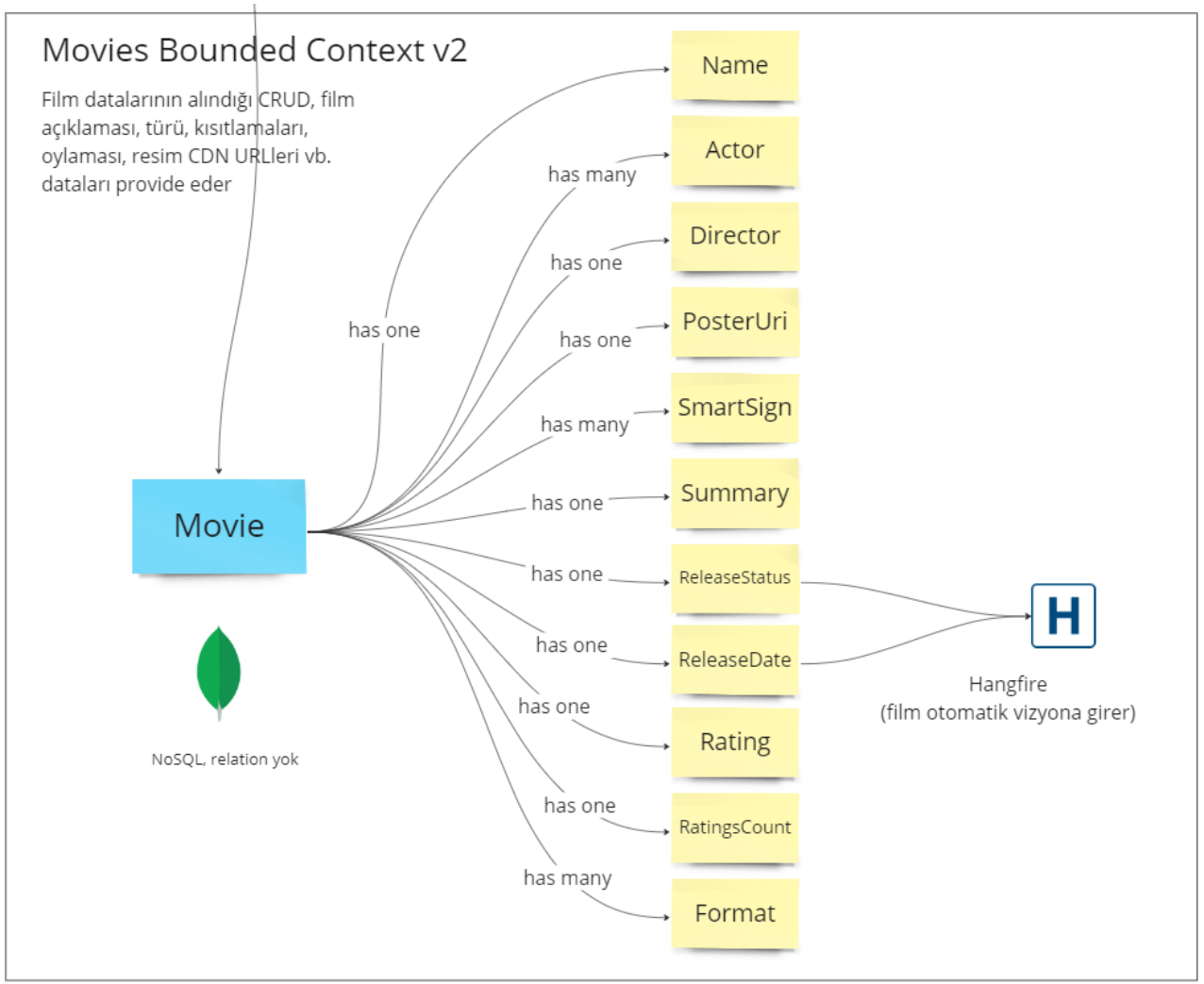
- Film metadataları Mongo ile BSON halinde tutulur
- Bu metadatalar validasyonlarda kullanılır (yaş)
- Filmlerin 3 durumu var: coming soon, vizyonda, ve released.
- Filmler vizyona çıkmadan önce çıkış tarihinde durumu vizyonda olması için programlanabilir. (Hangfire jobı)
- Kullanıcı bazlı film oylaması yapılır. Bu her kullanıcı 1 kere oy verebilecek şekilde rate limitlenmiştir.


```
_id: BinData(3, 'tO5xogJWskCx0x34E7fubg==')
Name: "Oppenheimer"
Director: "Christopher Nolan"
▸ Actors: Array
Genre: 19
Summary: "Oppenheimer, Amerikalı fizikçi Julius Robert Oppenheimer'ın hayatına o…"
PosterImageUri: "https://pbs.twimg.com/media/FvUVt3hXgAAxP1H?format=webp"
Rating: 7.5
RatingsCount: 2
▸ SmartSigns: Array
▸ Formats: Array
ReleaseStatus: 1
ReleaseDate: 2023-07-30T16:20:13.051+00:00
```

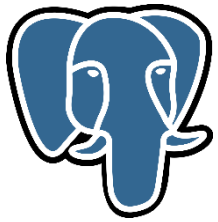
MovieService

GET	/movies	Get all movies.
POST	/movies	Create a new Movie entity.
GET	/movies/{movieId}	Get a Movie entity by its ID.
PUT	/movies/{movieId}	Update a Movie entity of given ID.
DELETE	/movies/{movieId}	Delete a Movie entity of given ID.
PUT	/movies/{movieId}/upcoming	Set the movie as upcoming, i.e. as an unreleased movie with determined release date in the future. At releaseDate, this movie is planned to be in the release status `ReleaseStatus.InTheaters`.
PUT	/movies/{movieId}/in-theaters	Set the movie as in theaters, i.e. as a recently released movie.
PUT	/movies/{movieId}/released	Set the movie as released, i.e. as a movie that has been released a long time ago.
PUT	/movies/{movieId}/release-date	Update the movie's release date to a given date.
PUT	/movies/{movieId}/rate/{rating}	Rate a movie with given ID, with the provided score.

MovieService



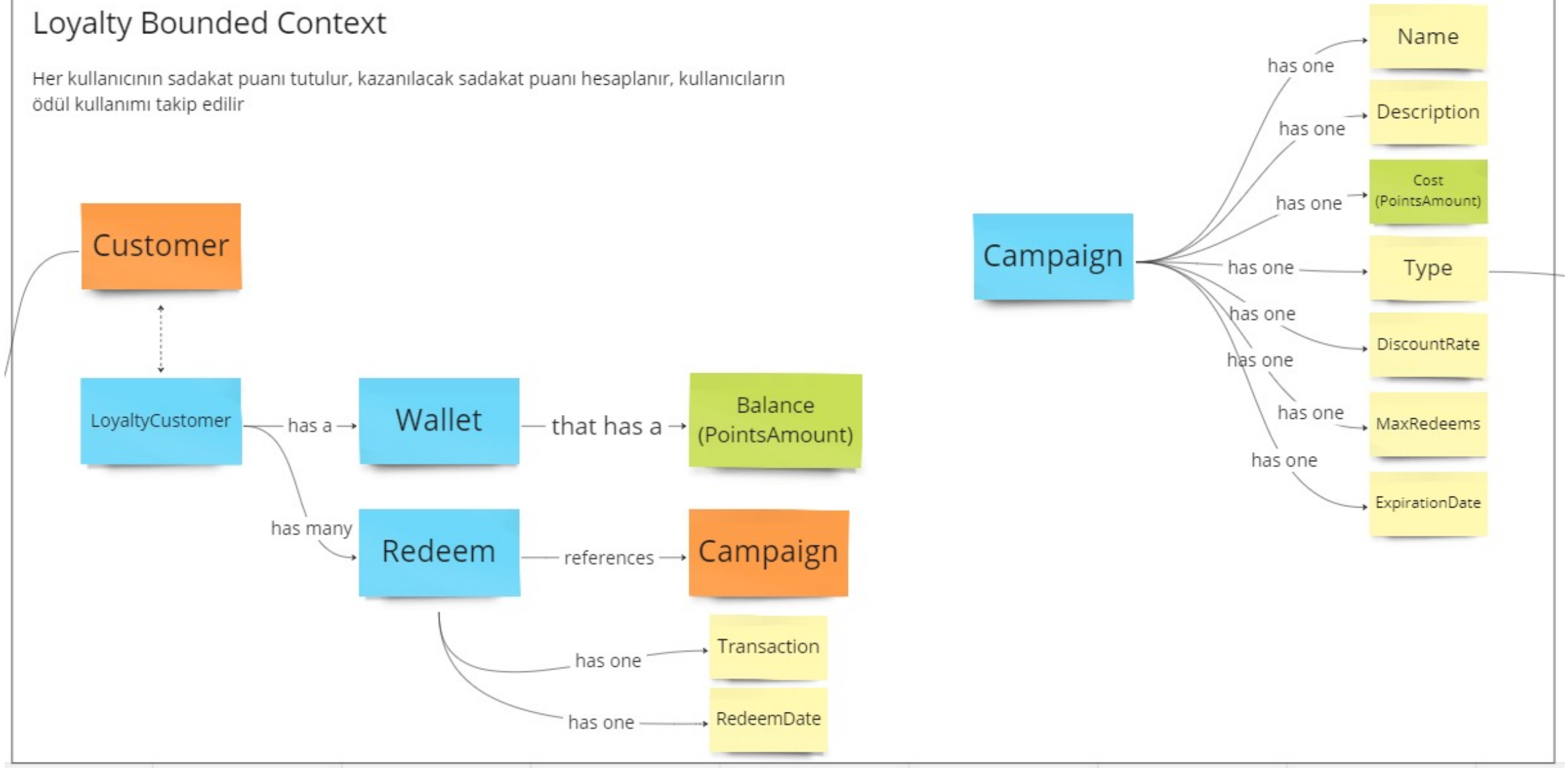
LoyaltyService



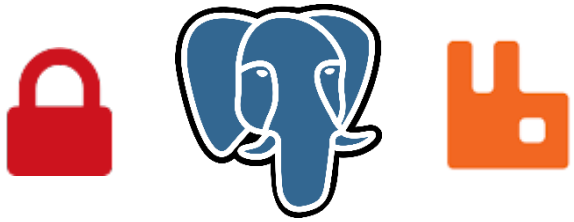
- Kampanyalar ve kullanıcı puan walletları yönetilir
- Kullanıcılar puanlarıyla bilet satın alırken kampanya alırlar
- Her bilet alımında belli bir yüzde puan kazanılır
- Kullanıcı IdentityService'e kayıt olduğu zaman Message Queue ile burada da aynı kullanıcıya karşılık bir loyalty kullanıcısı ve walletı oluşturulur
- Veriler Postgres'de tutulur. Kampanyalar update edildiğinde kullanıcılar clientlarındaki eski kampanya listesi ile kampanya almasın diye versioning ile optimistik concurrency kontrolü yapılır

Loyalty Bounded Context

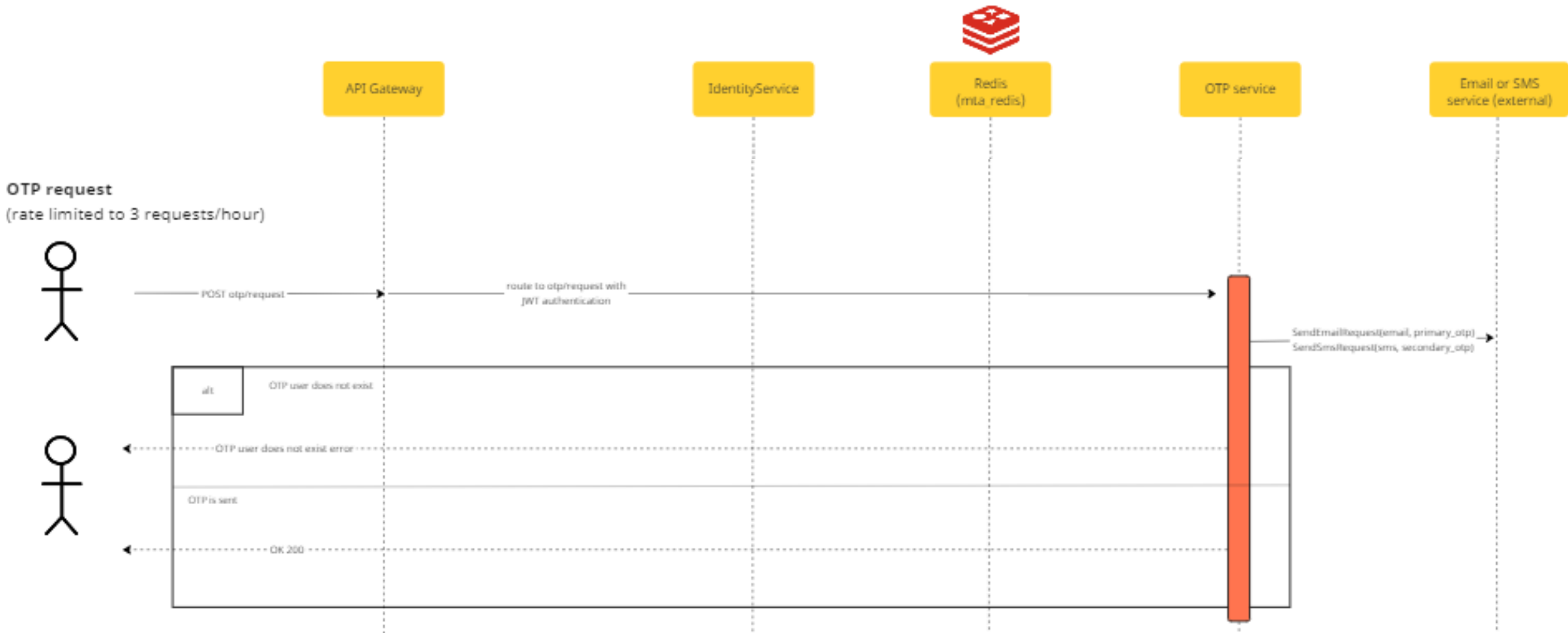
Her kullanıcının sadakat puanı tutulur, kazanılacak sadakat puanı hesaplanır, kullanıcıların ödül kullanımı takip edilir



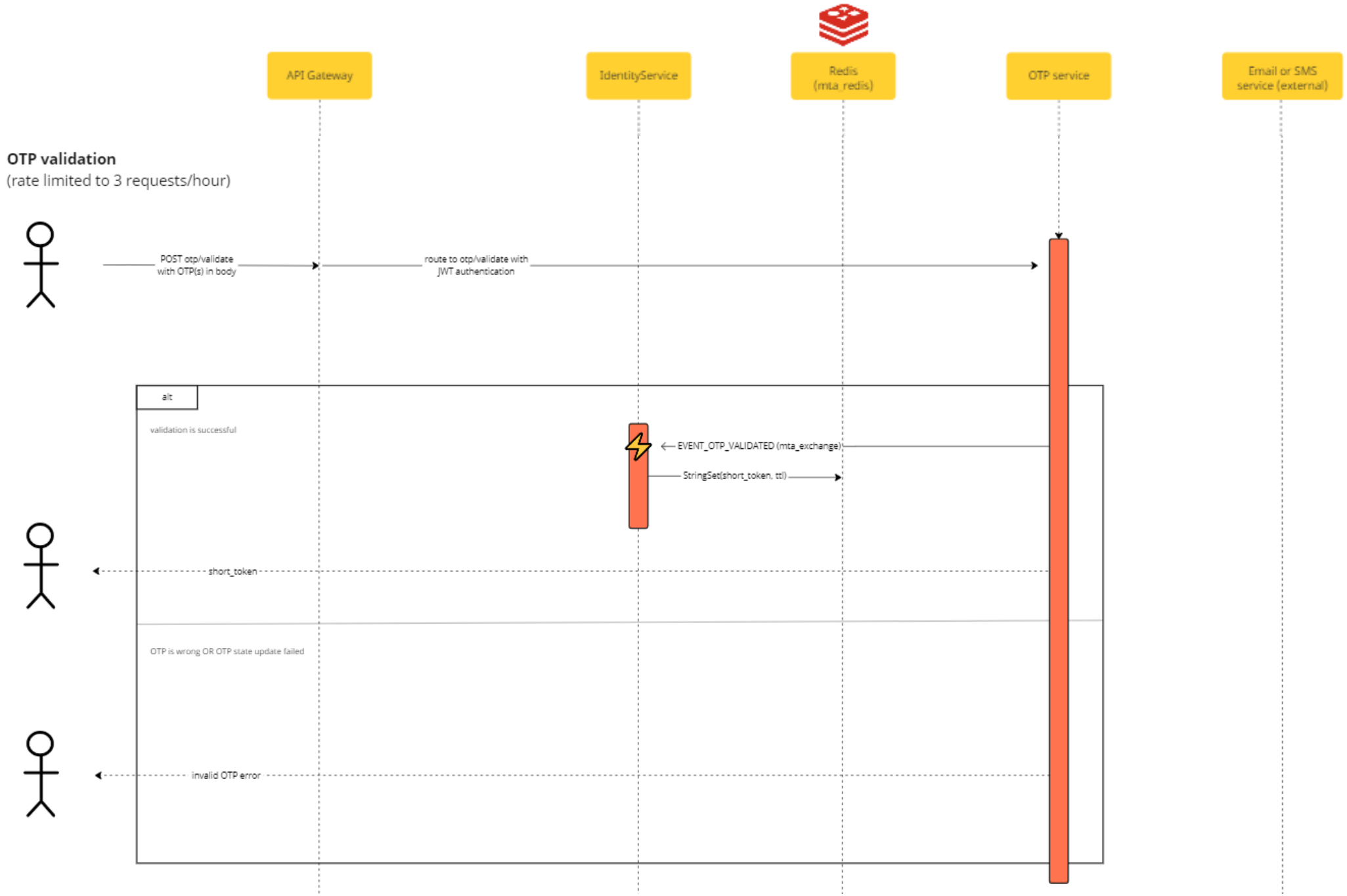
OTPService



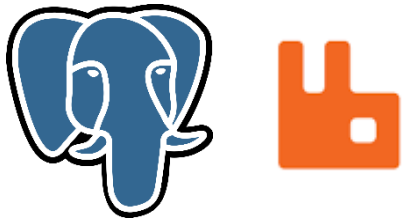
- OTP yönetimi ve validasyonu sağlanır. RFC4226 standartlı HOTP algoritması kullanılır. Bunun için Otp.Net kütüphanesini kullandım.
- Kullanıcıların OTP secretları ve counterları Postgres'te tutulur.
- IdentityService'ta üye olan her kullanıcı için message queue aracılığıyla burada da bir row (counter ve secret) oluşturulur.
- OTP'ler IdService'in Userinfo endpointinden alınan claimlere gönderilir.
- Saatte 3 request ve validasyon sağlanacak şekilde rate limitlenmiştir (API gateway tarafında).


API Gateway
IdentityService
Redis
(mta_redis)
OTP service
Email or SMS
service (external)
OTP request
(rate limited to 3 requests/hour)
POST otp/request
route to otp/request with
JWT authentication
SendEmailRequest(email, primary_otp)
SendSmsRequest(sms, secondary_otp)
alt
OTP user does not exist
OTP user does not exist error
OTP is sent
OK 200

OTP validation
(rate limited to 3 requests/hour)



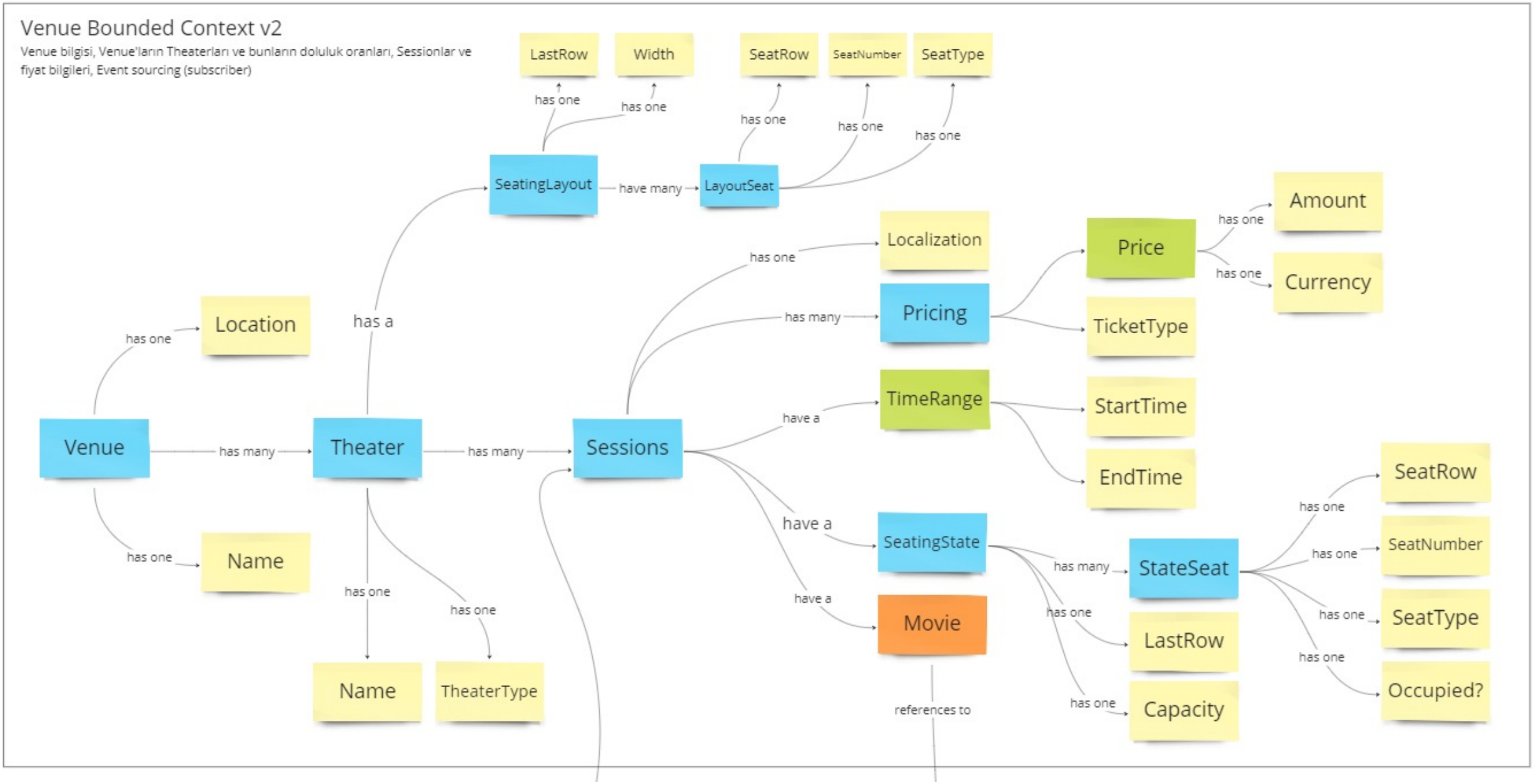
VenueService



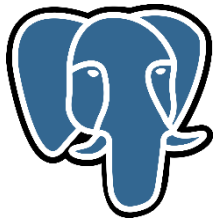
- Tüm sinema mekanları, bunlardaki salonlar, bu salonların koltuk düzenleri, bu salonlardaki film seansları ve bu seansların koltuk doluluk oranları yönetilir.
- Her salonun bir Layout'ı var. O salonda bir Session oluşturulduğu zaman, Layout ile seansa ait Seat objeleri oluşturuluyor. Bu Seat objeleri koltukların dolu olup olmadığını flagini tutuyor.
- Seat objelerinin her birinin concurrency token'ı var ve versioning ile concurrency kontrolü sağlanıyor.
- Seansların fiyatlandırması da bu serviste yapılıyor

Venue Bounded Context v2

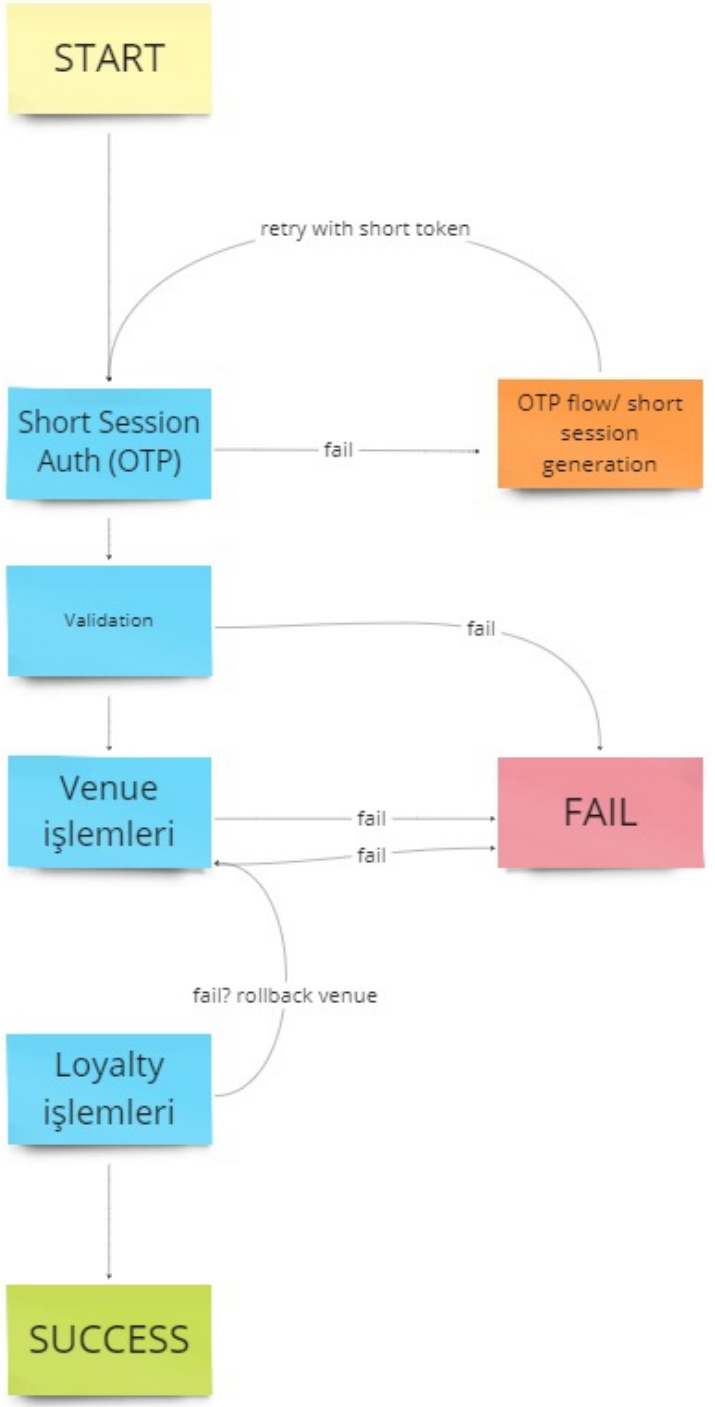
Venue bilgisi, Venue'ların Theaterları ve bunların doluluk oranları, Sessionlar ve fiyat bilgileri, Event sourcing (subscriber)

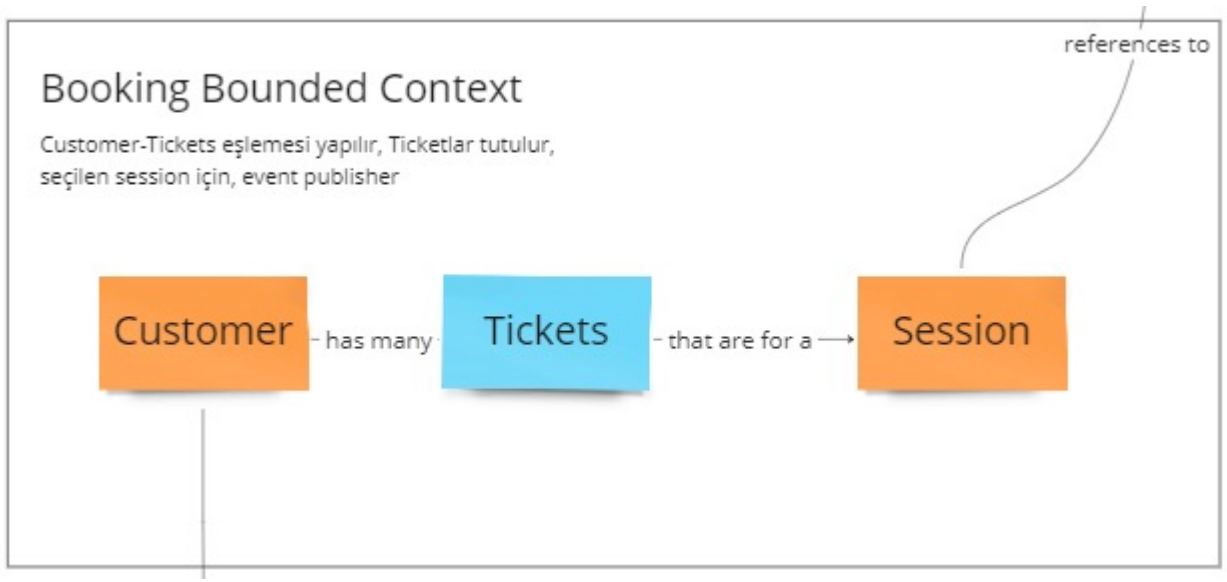


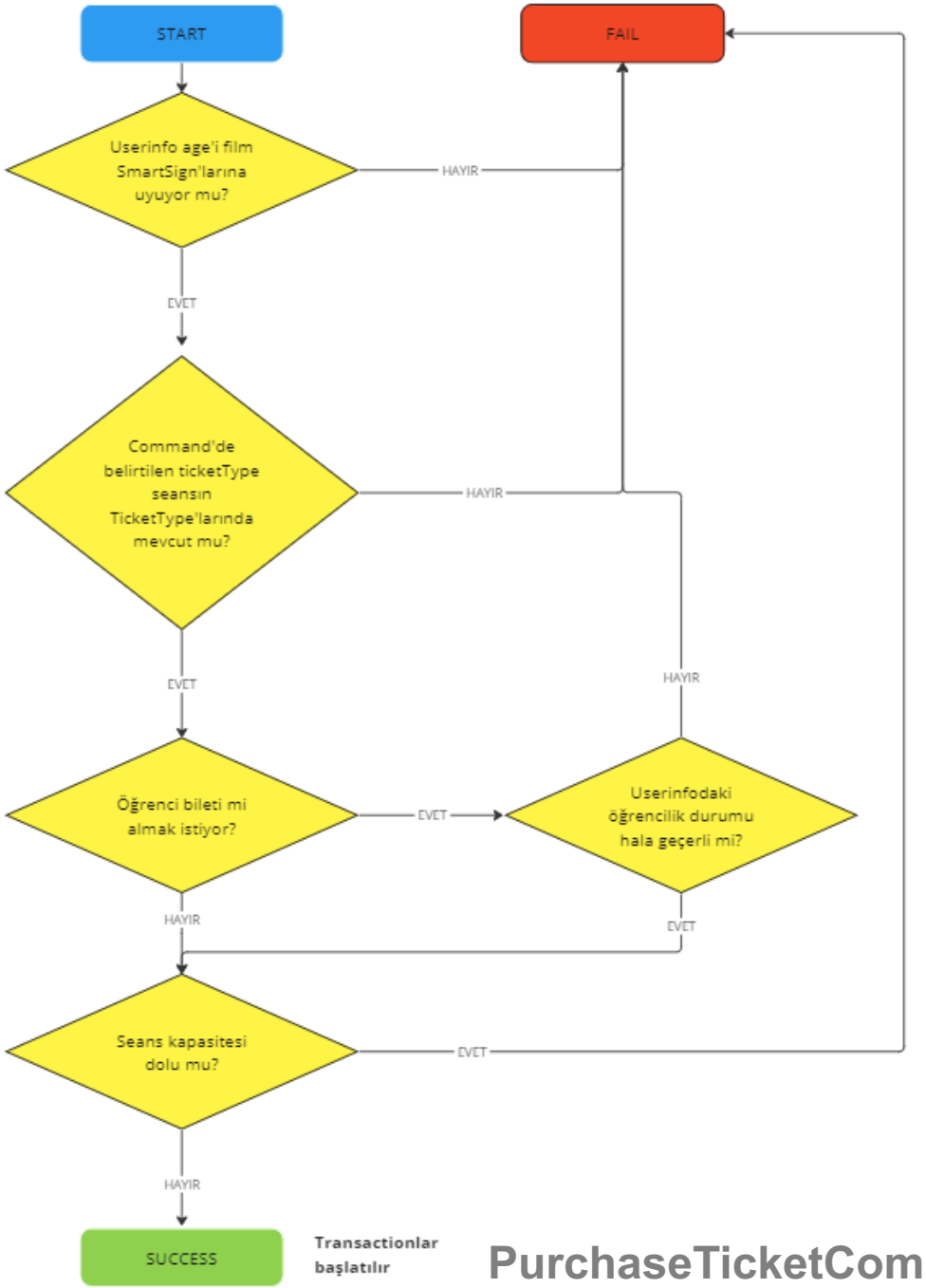
BookingService



- Bilet alma transactionını sağlar ve bilet verisini tutar.
- Diğer servislere HTTP request ve RMQ message göndererek validasyonlar ve işlemler yapar.
- İşlemlerden birinde hata gerçekleşirse compensating actionlar ile rollback yapılır.







PurchaseTicketCommand Validasyonları

PurchaseTicketCommand sequence diagramı Miro’da

Eksikler ☹

- Refactor gerek (kod çok karmaşık, çoğu Exception handlelanmıyor, daha iyi design patternlar kullanılabilir)
- Multifactor Authentication ayarlama akışı eksik
- Öğrencilik durumu açma akışı eksik
- Frontend arayüz eksik
- Daha çok test ve otomatize testler yapılmalı (yüzeysel ve manuel yapıldı)
- Resilience eksik (retrylar, compensating actionlar fail olursa ne olacak, graceful degradation, timeout…)
- Logging eksik
- Application layer'da CQRS ayrımı yapmadım, read modelde ORM kullanımını kaldırmalıyım

Postman demo

Neler öğrendim 

- EF Core modelleme, migrationlar, Code-First, concurrency control, versioning/concurrency token, psql ile serialized row locking, Lazy Loading Proxy, entity lifecycle
- Redis data tipleri, Redis .NET API'si, Mongo .NET driver'ı
- Anemic domain modellerden kaçınma
- DI ve scoped services
- RabbitMQ ile asenkron message iletişimi
- OIDC spesifikasyonu, IdentityServer, OpenIddict, Keycloak, OAuth2 flowları
- Ocelot middleware pipeline'ı ve DelegateHandler'lar
- MediatR ile App Layer – API layer arası CQRS command/handler iletişimi
- Fixed/Exponential/Jitter backoff'lu retry'lar
- SwaggerGen ve OpenAPI
- Hangfire, Serilog
- ProblemDetails standardı, Hellang ProblemDetails middleware'i
- Xunit
- HOTP ve TOTP algoritma spesifikasyonu

Teşekkür ederim ☺