|  |  |  |
| --- | --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ**  **İŞYERİ UYGULAMASI RAPORU** | Karadeniz_Teknik_Üniversitesi.gif |

|  |  |  |
| --- | --- | --- |
| **SONUÇ RAPORU**  **AD SOYAD**  **NO** |  | images.jpg |

|  |  |  |
| --- | --- | --- |
| **ÖĞRENCİ** | **Adı Soyadı** |  |
| **Öğrenci Numarası** |  |
| **T.C. Kimlik No** |  |
| **Bölümü** | Yazılım Mühendisliği |
| **Yarıyılı** | 20.. - 20.. Eğitim ve Öğretim ……. Yarıyılı |
| **İşyeri Eğitimi Süresi** | .../…/20.. - …/…/20.. ( ……. Gün) |
| **DENETÇİ** | **Adı Soyadı** |  |
| **Unvanı** |  |
| **Bölümü** | Yazılım Mühendisliği |
| **Adresi** | KTÜ Of Teknoloji Fakültesi |
| **İli** | Of/TRABZON |
| **İŞYERİ**  **YETKİLİSİ** | **Adı Soyadı** |  |
| **İşyeri Yetkilisi - Unvanı** |  |
| **İşyeri Bölümü** |  |
| **İşyeri/Firma Adı** |  |
| **İşyeri Adresi** |  |
| **İşyeri İli** | TRABZON |

|  |  |  |
| --- | --- | --- |
| **İŞYERİ EĞİTİMİ KOMİSYON BAŞKANI** | **DENETÇİ ÖĞRETİM ÜYESİ** | **İŞYERİ YETKİLİSİ** |
| Dr. Eyüp GEDİKLİ |  |  |
| / /20.. | / /20.. | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |
| --- |
| **YAPILAN İŞ: Uygulamada Refactoring ve Testlerin yapılması.** |

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

Uygulamada bulanan Dal modülünde ve içerisinde bulunan sınıflarda çok fazla iş yükü bulunmaktadır. Bu modülün mantıksal ve işleme bağlı kohezyon derecesi yüksek, sınıflar tek sorumluluk prensibine aykırı yazılmıştı. Dal modülünü bölmek için, içerisinde bulunan classların bağımlılıklarının iyice anlaşılması gerekiyordu. Grup toplantısı yapılarak, modülün uml diyagramının sadece modül içi bağımlılıklar gösterilecek şekilde çıkarılmasına karar verildi. Bu görev üstlenilerek dal mödülünü inceleyerek uml diyagramı hazırlanmaya başladı. Hazırlanan uml diyagramının bir kısmı bu bölümün sonunda bulunmaktadır. Hazırlanan uml diyagramı üzerinde Muzaffer bey ile toplantı yapıldı. Bu toplantıda dal modülü ve içerisindeki sınıfları nasıl böleceğimiz konusu tartışıldı. Haftaya Çarşamba günü proje yöneticilerine sunum yapılacağından zaman kısıttı bulunmaktaydı. Yetişebilmesi için kabul edilebilecek en uygun çözüm üzerinde anlaşıldı. Üzerinde anlaşılan çözüm ise;

* Dal modülü 3 modüle bölünecek.
* A modülünün görevi mesaj göndermektir. Buna göre çeşitli mesaj gönderme işlevlerini esnek biçimde desteklemesi gerekmektedir. Bunlara örnek UDP, TCP, USB vs. Tasarımında abstract factory pattern kullanılması kararlaştırıldı.
* B modülünün görevi sadece veri tutmaktır. Gönderilen veya gelen mesajların kuyrukları burada tutulacaktır. Tasarımında singleton pattern kullanılması kararlaştırıldı.
* C modülünün görevi A ve B modülleri için bir katman oluşturmaktır. Belirlenen portu devamlı dinleyerek gelen mesajları B modülünde bulunan kuyruğa eklerken, gönderilmesi gereken mesajları A modülünü kullanarak gönderir. Tasarımında facade pattern kullanılması kararlaştırıldı.

Belirlenen yapının kodlanması haftaya bırakıldı. Modül ve sınıflarda yapılan değişiklikler, entitylere eklenen yeni özelliklerin raporların gönderilmesinde ezilmeye neden olup olmadığının kontrolü için benimde dâhil olduğum 3 ekip üyesiyle test hazırlıklarına başlandı. Veri tabanının son halinin backup’u alınarak çalışma şartlarının eşit olması sağlandı. Gerekli configuration ayarları yapıldıktan sonra teste başlandı.

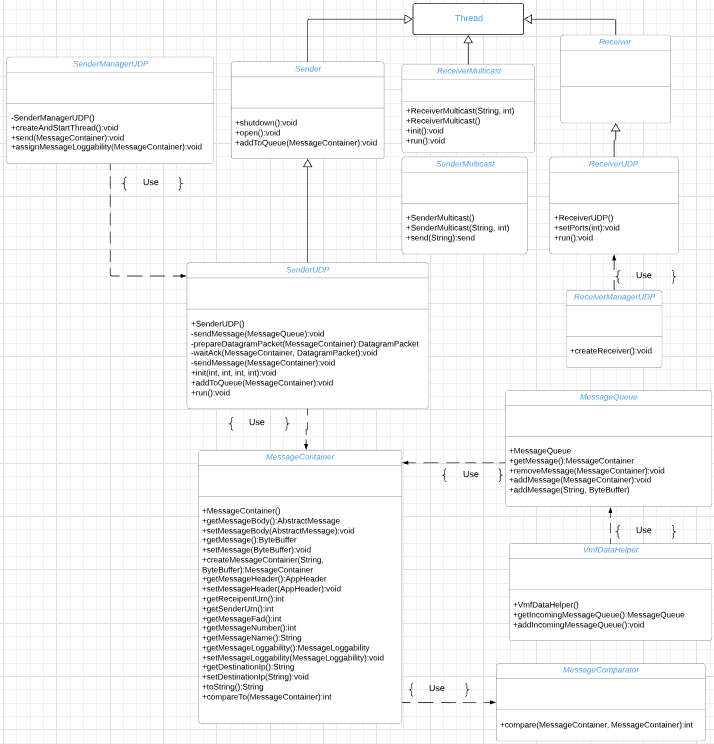
Test çalışması 128 entitynin sahip oldukları çizim şekillerinin tümünün sahip oldukları özellikler ile birlikte ağ ortamında gidip gelmeleri durumunda ortaya çıkan hatalar not alınarak çalışma haftası bitirildi.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |



|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |



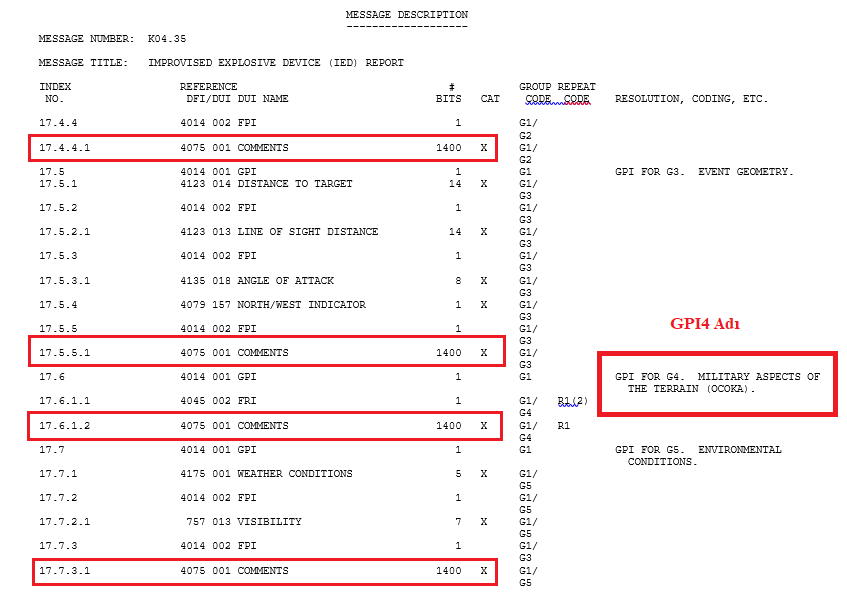
|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |
| --- |
| **YAPILAN İŞ: Bulunan Hataların düzeltilmesi, Test çalışmalarına devam edilmesi.**  **Proje Toplantısı ve Yeni görevlerin alınması** |

Geçen hafta yapılan testlerde karşılaşılan genel hata, raporlarda gitmesi gereken alanların gönderilmemesidir. Bu alanların başında comment gelmektedir. Comment alanlarının setlenme görevini üstlenildi. Projenin Git’e pushlanma sırasında ezilmelerin olduğu kararı verilip setlenmeyen alanlar için çalışmalara başlandı. Comment alanı için ilgili raporların sınıflarında createRaport fonksiyonlarında, rapor oluştururken comment field’inin raporda setlendiği ve tekrardan ilgili raporların sınıflarında saveReport fonksiyonlarında alınan rapor nesnesinde bu field’lerin dolu halde geldiği doğrulanıp, ilgili entity nesnelerinin niteliklerinin setlenme işlemi yapıldı.

IED raporunun comment alanının setlenmesi için rapor sınıfı incelenmeye başlandı. Her raporun en az bir comment alanı bulunmaktadır. IED raporunda birden fazla kullanıma özel farklı comment niteliği bulunmaktadır. Bahsedilen comment niteliğinin bir kısmı aşağıda görülmektedir.



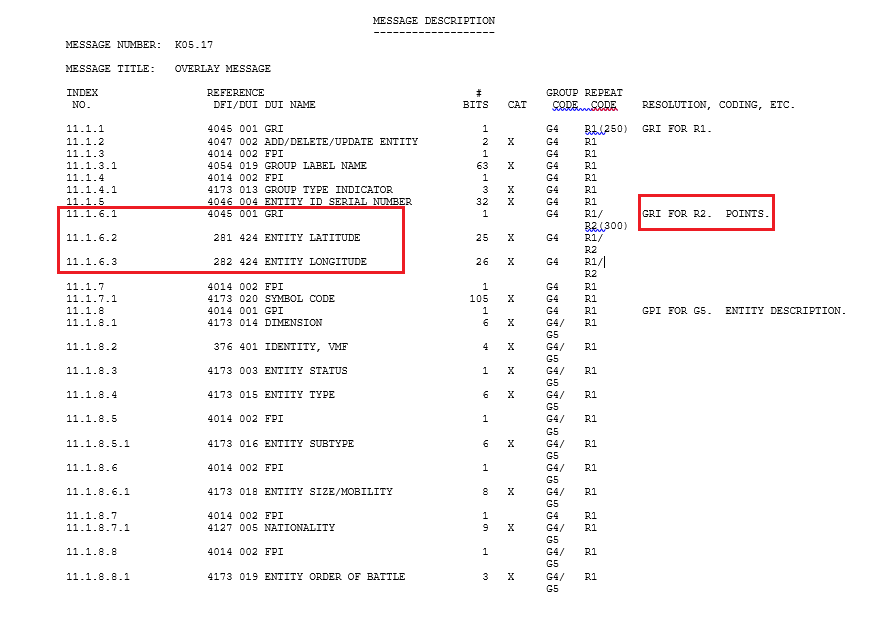
|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Kullanıma özel comment niteliklerinin adlandırılmasında genellikle GPI sınıflandırıcısı kullanılmaktadır. Buna örnek olarak bir önceki sayfada verilen 17.6.1.2 index numarasına sahip comment niteliği GPI\_4’e dâhil olduğundan adlandırılması MilitaryAspectsComment şeklinde olabilir.

IED rapor sınıfı incelenirken problem ile karşılaşıldı. Problem comment niteliğinin verilen örnekteki gibi yapılacağına, bazı comment nitelikleri için ayrı ayrı sınıf açılmıştır. Sınıf bildirgeci GRI olduğu için, sınıfın neden bu şekilde yazıldığı anlaşılamadı. Veri tabanları kontrol edilip FRI alanının GRI olarak alındığı doğrulandı. IED sınıfı çok önceden yazıldığından bu alanların şimdilik değiştirilmemesi gerektiği kararı alınıp vazgeçildi. IED raporunun 19.1 index numarasına sahip lastComment niteliği yorum için setlenmek üzere seçilerek problem şimdilik çözüldü.

Bir başka üstlendiğim sorun Overlay Message raporudur. Karşılaşılan soruna ilişkin VMF sayfası aşağıda görülmektedir.



|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Overlay Message raporu layere eklenip kaydedilen entityleri kapsülleyerek alıcılara göndermektedir. Entityleri kapsüllemek için Overlay Message sınıfının içerisinde Graphics sınıfı bulunmaktadır. Bu sınıfın bazı nitelikleri bir önceki sayfada verilen resimde görülmektedir. İşaretlenen Point sınıfı ise entity çizim çizimlerini desteklemek için bulunur. Çizim biçimi Multi Line veya Polygon olan entityler çok noktalı oldukları için Graphics içerisinde ayrı bir sınıf olup liste şeklinde tutulmaktadır.

Overlay raporunda karşılaşılan sorun, Layere 5’den fazla entity eklendiğinde eklenen entitylerden sadece bazılarının gitmesidir. Gönderilen entitylerin alınma sayısında değişiklik olup olmadığının kontrolü için tekrardan test yapıldı. Yapılan testin sonucunda alıcı tarafa maksimum 4 entity iletildiği belirlenip doğrulandı.

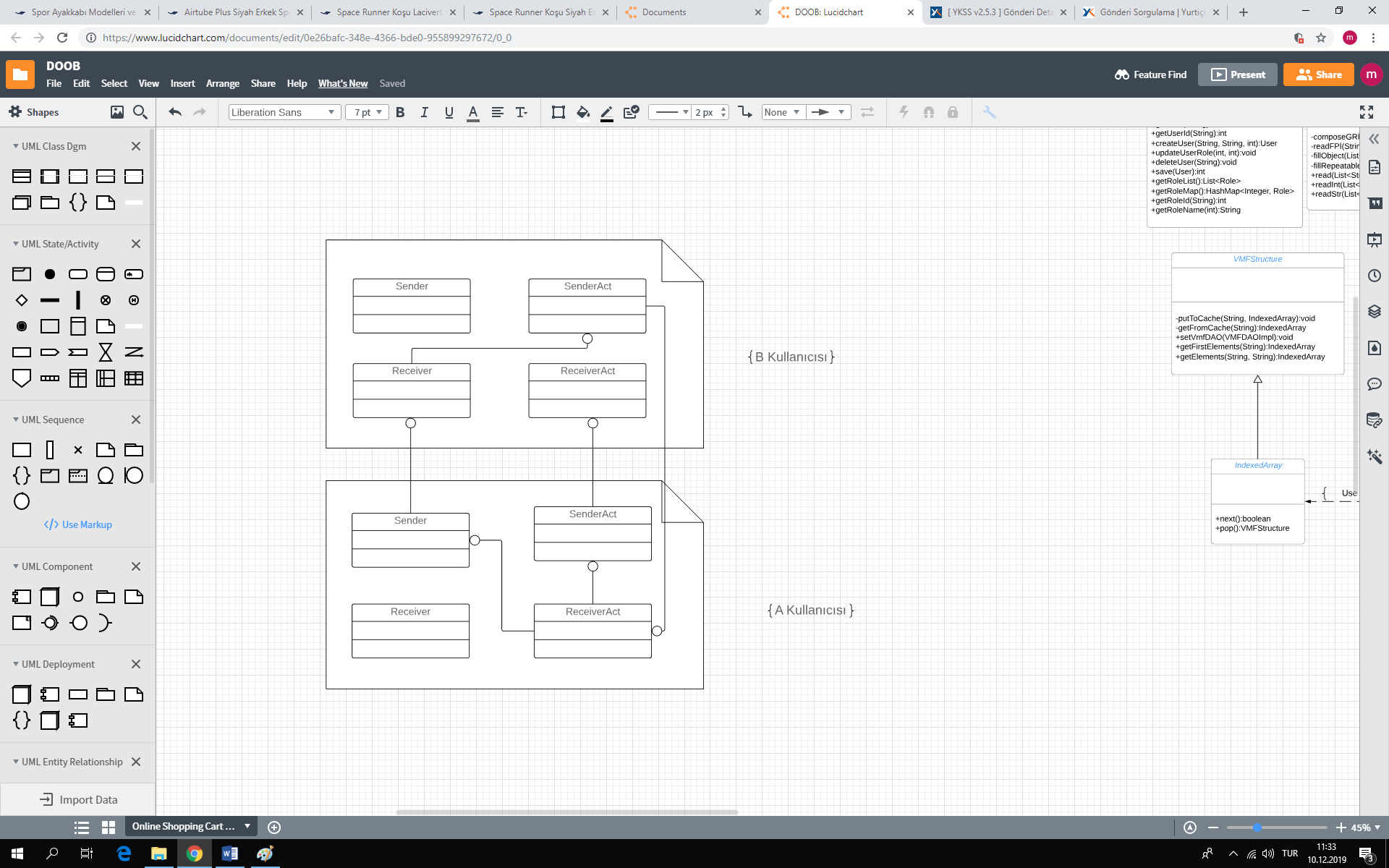
Çalışmalara kendi bilgisayarımda başladım. Kendime entity göndermek için uygulama çalıştırılarak haritaya bir entity eklendi. Uygulama kapatılıp veri tabanında çeşitli değişiklikler yapılarak, eklenen entitynin uygulama için farklı bir entity olarak algılanması sağlandı. İlgili entitye Overlay raporu açıldı. İlk olarak oluşturulan raporun bitlere dönüştürüldüğü VMFMessageBodyComposer2 ve alıcı tarafta gelen bitlerin rapora dönüştürüldüğü VMFMessageBodyParser sınıflarının metotları incelenmeye başlandı. Bir sorunla karşılaşılmayınca OverlayReportServiceImpl sınıfında alıcı tarafında saveMessage fonksiyonu, gönderici tarafında ise createMessage fonksiyonları incelendi. SaveMessage sınıfında, haritaya eklenen entitylerin hepsi Graphic nesnesi olarak rapora ekleniyordu. Alıcı tarafında Graphic listesinde 4 adet eleman bulunuyordu. Verilerin alınmasında ReceiverUDP sınıfı kullanılmaktadır. Bu sınıfın incelenmesine karar verildi. Yapılan incelemede gelen veri uzunluğunda bir kısıt konulmuştu. Bu kısıt göndericiden gelen belli veri paketinden belli sayıda bit alınmasına neden oluyordu. Kısıt sınırı belirli bir sınıra çıkarılarak 10 entity sınırı getirildi. Telsizlerde haberleşme yapılacağından çok büyük veri paketlerinin gönderilmesi tercih edilen bir şey değildir. Pakette bir hata oluştuğunda, tüm paketin tekrardan alınması gerekmekte ve bu durumu çok fazla birim için düşündüğümüzde ağ ortamında bir yoğunluk yaratacaktır. 10 entity sınırının ilerleyen zamanlarda değiştirilmesi gerektiği not alındı. Yapılması gereken, entitylerin 2 3’erli gruplar halinde Overlay göndermektir.

Muzaffer bey ile daha önceden konuşulan UDP işlemleri ile ilgili tekrardan toplantı yapıldı. Yapılan toplantı neticesinde UDP işlemlerinin bir sonraki sayfada verilen resimde görülen yapıda – benzer olması kararlaştırıldı.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Resimde A kullanıcısı bir raporu B kullanıcısına gönderme işleminde sınıfların bağımlılıklarını göstermektedir. Projede bu değişikliğe bu hafta yapılacak toplantı sonrasında başlanmasına karar verildi.



Muzaffer bey ile toplantı yapılırken diğer çalışma arkadaşlarımız testlere devam etmekteydi. Teste devam edilirken Tasks&Events bölümünde bulunan KIDNAPING entitynin polygon biçimi için çok noktalı (13 nokta) çizilmesini istedim. Çok noktalı çiziminde bir sorun ile karşılaşılmadı. Aynı entity güncellendiğinde veri tabanından hata alındı. Veri tabanı kontrol edildiğinde çizilen KIDNAPING entity’nin tek olması gerekirken çiftlenmişti. Tüm entitylerin veri tabanında tekil olması gerekmektedir. Bu hata durumunun anlaşılması için aşağıdaki adımlar uygulandı;

* Belli bir nokta aralığında olup olmadığının kontrolü için değişken boyutlarda çizimlerin yapılması
* İlgili entity için oluşturulacak raporun açık veya kapalı durumlarında çizimlerin yapılması
* Bu durumun sadece KIDNAPING entitysine özgü olup olmadığının kontrolü.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Yapılan uzun çalışmaların sonucunda ilk başta ilgili entity için oluşturulan rapordan kaynaklı olduğu düşünülse de bu düşüncenin yanlış olduğu anlaşıldı. Hatanın oluşumunda tekrar eden bir stabil durum yakalanamamaktadır. Hatanın anlaşılması için çalışmalara kendi bilgisayarımda başladım. Kendime entity göndermek için uygulama çalıştırılarak haritaya bir entity eklendi. Uygulama kapatılıp veri tabanında çeşitli değişiklikler yapılarak, eklenen entitynin uygulama için farklı bir entity olarak algılanması sağlandı. İlgili rapor açılarak uygulama debug ile kontrol edilmeye başlandı. Daha önceden yazdığım StageEntity sınıfında, KIDNAPING entitysinin detay ekranını açarken bir sorun yakaladım. Bu sınıf detay ekranı açılacak entityi bir HashMap’tan çekmektedir. Bu HashMap projenin başlangıcında oluşturulmuş ve projenin en alt seviyelerinde kullanılmaktadır. Bundan dolayı bu Mapin nerede ne şekilde kullanıldığı hakkında fazla bilgimiz bulunmamaktadır. Yakalanan sorun ise bu mapten gelen entity’nin idsinin farklı olmasıdır. Entity ilk oluşturulurken random bir id verilip, veri tabanına kayit edildiğinde veri tabanında otomatik yaratılan id ile güncellenmektedir. Hatanın bulunması için bahsedilen HashMap’ın kullanıldığı yerler projede aratılarak incelenmeye başlandı. Uzun bir incelemenin ardından AbstractLayerController sıfınında updateGraphic fonksiyonunda, bahsedilen HashMap’in güncellendiği kısım bulundu. Buradaki güncelleme Hasmap üzerindeki son işlemdir. Dolayısıyla buradan kaynaklanan hata palet ekranı açıldığında StageEntity sınıfında oluşan hatadır. Debug atılarak kodun işleyişi incelenmeye başlandı. Bir süre thread yüzünden oluşabilecek bir olay gözlemdi. AbstractLayerController sıfınında bulunan updateGraphic fonksiyonun işlemi bitmeden daha önceki incelemelerde debug koyulan EntityHelperImpl sınıfında bulunan saveOrUpdate fonksiyona geçiş yapılmaktadır. saveOrUpdate fonksiyonu haritaya entity eklendiğinde veya güncelleme yapıldığında veri tabanında işlemlerin yapıldığı fonksiyondur. Daha önceden bahsedilen entitylerin id’lerinin güncellenmesi burada yapılmaktadır. Fonksiyondaki bu işlem kesilmesine neden olan thread aranmaya başlandı. EntityUiService sınıfı hem EntityHelperImpl hemde AbstractLayerController sınıfında bulunan updateGraphic fonksiyonunu tetiklemektedir. AbstractLayerController sıfının tetiklenmesi thread içerisine alındığından işlemlerin sırayla yapılması engellenmiştir. İşlemler sırayla yapılamadığı içinde HashMap’a eklenen entity farklı id ile eklenmekte, bu durum hataya sebeb olmaktadır. Thread silinerek hata düzeltildi.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Perşembe günü 13.00’de toplantı yapılacağından dolayı proje hızlıca bilgisayarlarda test edildi. Projenin son halinin jar çıktısı alınarak tabletlere yüklendi. Telsizlerin kurulumu yapılarak tabletlerde test işlemi yapıldı. Bir sorunla karşılaşılmayınca gerekli dokümanlar cihazlar alınarak toplantı odasına gidildi. Toplantı için cihazlar ayarlanarak proje hazır hale getirildi. Projeye takılan ekip liderleri ve proje yöneticilerine sunum yapılarak projenin eksik olan yönleri belirlenip ilgili gereksinimler not alındı. Toplantı 16:00 bitirilip, cihazlar ve dokümanlar hazırlanarak çalışma yerine gelindi.

Toplantıda yapılması kararlaştırılan gereksinimler paylaşıldı. Aldığım gereksinim, harita üzerine eklenen ve sadece Taktik kullanıcılarına ait entitylerin bilgilerinin güncellenmemesi durumunda entitylerinin altlarında yeşil turuncu ve kırmızı renklerin olmasıdır fakat bu görev tmap ekibine ait olduğundan dolayı onlardan bu yapı gelene kadar opacity değerleriyle oynanacaktır. Süre olarak 0-30dk yeşil, 31 dk – 3 saat turuncu ve 3 saat üzeri kırmızıdır renklerine karşılık gelebilecek opacity değerleri belirlenecektir. Harita üzerinde 100 adet Taktik kullanıcı olduğu varsayılır ise bu entityleri kontrol edip, entity hakkında bilgi geldiğinde o entitye ait süreyi sıfırlayan bir yapının kurulması gerekmektedir. İlk olarak Thread kullanılması düşünülse de bundan vazgeçildi. Hazelcast’de cacheye koyulan datalara time listener atanabilmektedir. Timer atanan data, süresi dolduğu anda geri bildirim yapmaktadır. Bu yapının kullanılıp kullanılmayacağı araştırıldı. Projede spring cache kullanıldığından Hazelcastın bu özelliğine erişim olmamaktadır. Son fikir olarak Timer ve TimerTask yapısı kullanılmaya karar verildi.

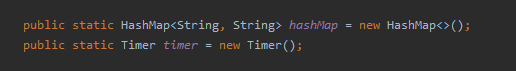
Soyut bir sınıf olan **TimerTask** **Runnable** arabirimi uygular ve dolayısı ile aslında bir [Java Thread](http://www.fatihkabakci.com/Makaleler-JAVA_THREAD_OLUSUMU) olarak çalışır. Bundan dolayı run() metodunu override ederek, gerçekleştirmek istediğiniz task kodunu, bu metoda yazılması gerekmektedir. TimerTask sınıfının diğer metotları arasında mevcut taskın işi bittiği zaman sonlandıran cancel() ve taskın ne zaman çalıştırılacağı bilgisini bulunduran scheduledExecutionTime() metotları yer alır.

Timer sınıfı somut olup, doğrudan örneklendirilebilen bir sınıftır. Genel olarak en önemli metotları schedule() 'dir. Bu metodun hem long tipinde, hem de bu değerlerin Date karşılıkları tipinde overload edilmiş metotları bulunur. Aralarındaki farklar zamanlayıcının ilgili taskı, ne zaman ve hangi aralıklarda çalıştırılması ile ilgilidir. Zaman değerleri milisaniye cinsinden tanımlanır.

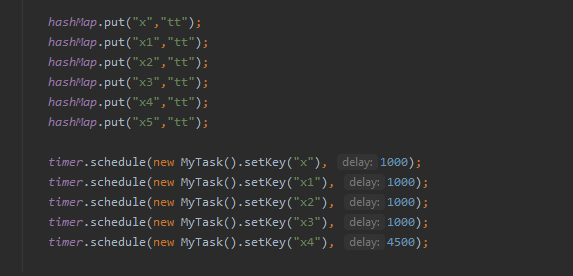
|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

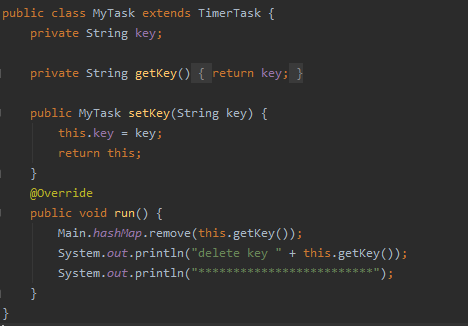
Timer yapısının nasıl kullanılacağı hakkında ön bir çalışma yapılmaya başlandı.



Main fonksiyonunda resimde verilen referanslar tanımlanmıştır.



Main fonksiyonunda tanımlanan Map referansına eklemeler yapılarak özel olarak oluşturulan Task yapısı içerisinde key numarası verilen data Map referansından silinecektir. MyTask sınıfı aşağıda görülmektedir.



Örneği yapılan uygulamanın benzeri projede uygulanacaktır. İlk düşünce Taskların entity tipinde niteliğin tanımlanmasıdır. Süresi dolan Taskların run metodu çalışacaktır. Burada kontrol edilmesi gereken ilk durum, Task içerisinde bulunan entity için Timer içinde beklerken güncelleme gelip gelmediğidir.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Bunun anlaşılması için raporların alınmasında Singleton Tasarımı ile oluşturulmuş bir sınıf içerisinde bulunan HashMap referansına gelen entity’nin urn si key, geldiği andaki sistemin saati ise value olacak şekilde HashMap referansında bulunuyorsa güncellenecek yoksa koyulacaktır. Task içerisindeki run metodu çalıştığında, Singleton yapısındaki HashMap referansı içerisinde tutulan data içerinde kendi urn’sine karşılık gelen son güncellenme bilgisine göre işlem yapılacaktır. Eğer timer içerisindeyken güncelleme geldiyse, ne kadar önce geldiğinin hesaplaması yapılarak, tekrardan timere koyulacaktır. Örneğin bir entity timer içerisindeki 25. Dakikasında bir güncelleme geldiyse, 30. Dakikada tast içerisinde bunun kontrolü yapılarak 25 dakika için tekrardan timere koyulacaktır. Eğer gelmediyse kendisini 31 dk – 3 saat aralığında koyacaktır. 31 dk – 3 saat aralığından çıkan task ise kendisini Singleton yapısındaki HashMaptan silecektir. Task içersinde timere tekrardan kendisi koyacağı için, Timer referansıda Singleton olacaktır. Bu durumda ilk düşünülen gerekli şartların kontrolü yapıldıktan sonra timer içerisine Task içerisinde this gösterceği ile direk kendisini timere koymasıdır. Fakat planlanan bu case’nin çalışmayacağı anlaşıldı.

Timer içerisinide koyulan Tasklar bir kuyrukta tutulmaktadır. Süresi dolan tasklar kuyruktan çekilerek state niteliği cancel yapılır. Bu nitelik TimerTask abstract sınıfında bulunmaktadır ve private olarak tanımlanmıştır. Timer içerisine eklenen Tastkların state özelleğinin cancel olmama şartı bulunduğundan, State özelliği cancel olarak değiştirilen Task Timer içirişine koyulamamaktadır. Bundan dolayı süresi tamamlanan Tasklar run metotlarında düştüğünde tekrardan Timer içerisine koyulması için run metodu içerisinde yeni bir Task oluşturup Task kopyalandıktan sonra Timere koyulacak yada reflection ile state niteliğinin erişim belirteci public yapılarak state niteliği değiştirilecektir. Açıklanan 2 durumdan kopyalanma tercih edilmiştir.

Yapmak için üstlendiğim gereksinimin değiştirilmesine karar verildi. Üstlendiğim yeni gereksinim, raporların kaydedilmesidir(Log işlemi). Bunun için gelen ve gönderilen raporların kayıtları tutulacak ve kullanıcı kayıt ekranını açtığında kayıt dokümanlarına erişecektir. Listelenen kayıtlar arasından seçilen bir kayıt olursa, ekranda bir panel açılarak ilgili raporun içeriği kullanıcıya gösterilecektir. Kayıtlar için filtre uygulanabilecektir; zamana veya zaman aralığına, kullanıcılara (kendisinin gönderdiği veya seçilen kişi/kişilerden gelenler), rapor tiplerine göre kayıtlar listelenebilecektir. Filtrede çoklu seçim desteklenecektir. Bırakacağım entity gereksinim görevini Büşra üstlenecektir. Üzerinde araştırmalar yaptığım entity gereksinimini, yazmış olduğum örnek uygulama üzerinden anlatarak kurguladığım yapı hakkında bilgi verilerek çalışma haftası bitirilmiştir.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |
| --- |
| **YAPILAN İŞ: Jira Üzerinde Storylerin yazılması ve Üstlenilen Gereksimin Kodlanmaya Başlanması** |

Projenin test ekibine teslimi yaklaştığı için jira üzerinde projenin Storyleri oluşturulmaya başlandı. Jira proje ve süreç yönetimi işlemleri için şirketlerin kullanmış olduğu pek çok uygulamanın başındakilerden birisidir. Kullanım kolaylığı, stabil oluşu ve Agile yöntemleri destekliyor olması, jira’yı bilhassa yazılım geliştirme dünyasının vazgeçilmez iş takibi uygulamaları arasında kılmaktadır. Tabi pek çok farklı sektörde de kullanıldığını söylemek gerekmektedir.

Jira ile yapılan işlemlerin bazıları;

* İşleri oluşturup ilgili kişilere atayabilir
* Oluşturulan işlerde ne kadar vakit ayrıldığını loglayabilir, estimation yapabilir
* Popüler Continuous Integration, Messaging, Version Control vb. uygulamalarla tamamen entegre çalışılabilir
* Planlama, sürüm yönetimi, gelişmiş raporlama gibi işlemleri çok rahat gerçekleştirebilir
* Scrum ya da Kanban yöntemleri uygulayabilir
* En önemlilerinden biri de kendi iş akışınızı (workflow) oluşturabilirsiniz.

Yazılan bazı Storyler;

1. **Palet Ekranından Poligon Sembol Çizimi**

Haritanın üst orta kısmında bulunan üçgen kısmına tıklanır. Açılan palet ekranından sembol seçildiğinde, seçilen sembolün etrafında çerçeve ve ekranın sağında 4 adet hostility tipi çıkar. Seçilen hostility tipine göre, hostility altında seçilen sembole uygun çizim biçimleri çıkar. Çizim biçimlerinden poligon seçilir. Poligon biçimi, dört köşesi kapalı dikdörtgen olan resimdir. Çizim biçimi seçildiğinde palet ekranı kapanır. Seçilen çizim biçimi poligon ise harita üzerinde kapalı bir alan işaretlenecektir. Bundan dolayı haritada en az üç nokta belirlenmelidir. Başlangıç noktası için harita üzerinde belirlenen/istenilen konuma basılır. Başlangıç noktasından sonra poligon için gerekli olan ikinci nokta tekrardan harita üzerinde belirlenen/istenilen konuma basılır. Seçilen iki noktadan sonra, harita üzerinde belirlenen/istenilen konuma basıldığında, otomatik olarak basılan konum ile başlangıç konumu birleşmektedir. Bu durum üç nokta için üçgeni oluşturmaktadır. Çizimi bitirmek için harita üzerinde belirlenen bitiş konumuna basılı tutulur.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

1. **Favori Ekranından Multiline Sembol Çizimi**

Haritanın sağ orta kısmında bulunan üçgen kısmına tıklanır. Açılan favori ekranından sembol seçildiğinde, seçilen sembolün etrafında çerçeve ve ekranın altında 4 adet hostility tipi çıkar. Seçilen hostility tipine göre, hostility altında seçilen sembole uygun çizim biçimleri çıkar. Çizim biçimlerinden multiline seçilir. Multiline biçimi, u biçiminde olup bir uçu açık olan resimdir. Çizim biçimi seçildiğinde favori ekranı kapanır. Harita üzerinde belirlenen/istenilen konuma basılır - line başlangıç noktası seçilir. Harita üzerinde belirlenen/istenilen noktalara tıklanarak line çizimine devam edilir. Line çizimini bitirmek için bitiş noktasından ekrana basılı tutulur.

1. **Palet Ekranının Temizlenmesi**

Palet ekranında sembol seçildiğinde, seçilen sembolün etrafında çerçeve ve ekranın sağında 4 adet hostility tipi çıkar. Seçilen hostility tipine göre, hostility altında seçilen sembole uygun çizim biçimleri çıkar. Palet ekranı sembol ve hostility tipi seçilip, kapatılırsa, palet ekranı tekrardan açıldığında yapılan seçimler silinmiş olacak ve ekranda sadece semboller gözükecektir. Palet ekranının sağ altında bulunan Clear Selection butonuna tıklandığında da yapılan seçimler silinmiş olacak ve ekranda sadece semboller gözükecektir.

1. **Palet Ekranından Point Sembol Çizimi**

Haritanın sağ orta kısmında bulunan üçgen kısmına tıklanır. Açılan favori ekranından sembol seçildiğinde, seçilen sembolün etrafında çerçeve ve ekranın altında 4 adet hostility tipi çıkar. Seçilen hostility tipine göre, hostility altında seçilen sembole uygun çizim biçimleri çıkar. Çizim biçimlerinden point seçilir. Point biçimi, içerisinde sadece siyah küçük daire olan resimdir. Çizim biçimi seçildiğinde favori ekranı kapanır. Harita üzerinde belirlenen/istenilen konuma basılır.

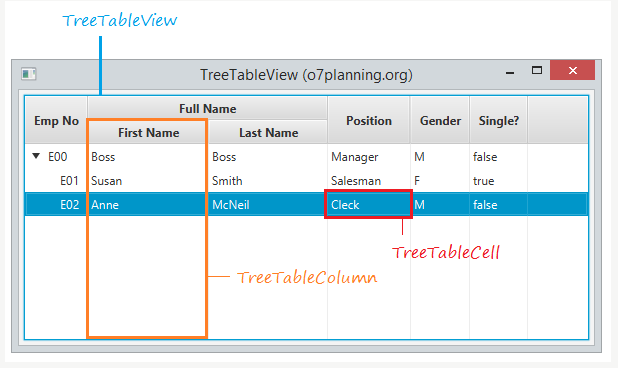
1. **Favori Ekranından Daire Sembol Çizimi**

Haritanın sağ orta kısmında bulunan üçgen kısmına tıklanır. Açılan favori ekranından sembol seçildiğinde, seçilen sembolün etrafında çerçeve ve ekranın altında 4 adet hostility tipi çıkar. Seçilen hostility tipine göre, hostility altında seçilen sembole uygun çizim biçimleri çıkar. Çizim biçimlerinden point seçilir. Point biçimi, içerisinde sadece siyah küçük daire olan resimdir. Çizim biçimi seçildiğinde favori ekranı kapanır. Harita üzerinde belirlenen/istenilen konuma basılır.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Raporların kayıtlanmasında, gelen ve gönderilen raporların uygulamada kayıt ekranında gösterilip veri tabanına kayıt edilecektir. Kayıt ekranının esnek tasarımı için araştırmalara başlandı. JFoeniX kütüphanesinde bulunan ve aşağıda resmi verilen Tree Table View’in kullanılmasına karar verildi.



Tree Table View ağaç yapısını kullanmaktadır. Resimde verilen Emp No column’una bakılırsa bu yapı görülmektedir. Eklenen veriler iç içe geçirilebilmektedir. Column’lara tıklandığında otomatik tüm veriler ASC veya DESC olarak sıralanmaktadır. Gereksinimin yapılmasında zaman kısıtı olduğu için ön deneme yapılmadan projede kodlanmaya başlandı.

Kayıt ekranının açılmasının nereden tetikleneceği ve projenin ara yüzünde nerede gösterileceği konusunda karar verildi. Kayıt ekranı ekranın sağ altında bulunacaktır. Yüksekliği, ekran yüksekliğinin 0.40’ı ve ekran genişliğinin 0.31’i olmasına karar verildi. Kayıt ekranı 3 kolondan oluşacaktır. Bu kolonlar, Mesajı oluşturan kişinin LongName’si, raporun iletilme veya oluşturulma tarihi ve rapor tiplerini ifade etmektedir. Ekranın açılması ise ekranın alt kısmından yapılacaktır. Bunun için geçici olarak bir resim eklenecektir.

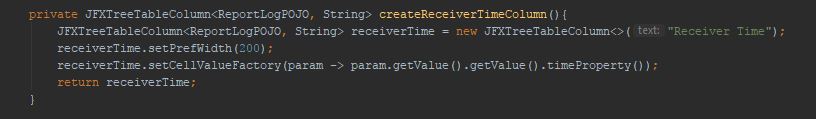
Kayıt ekranı için LogScreen.fxml adında dosya oluşturulmuştur. İçerisine sadece JFXTreeTableView tanımlanarak fx:id’si verilmiştir.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

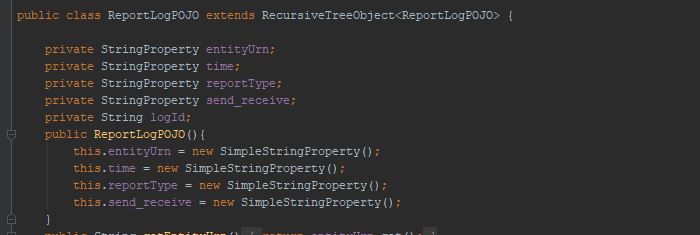
|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

Oluşturulan fxml dosyası için controller sınıfı oluşturularak adına LogScreenController verilip sınıf sprinde devredilmiştir. Log ekranın açılması için ekranın alt kısmında bulunan arayüze belirlenen resim eklenerek click özelliği verilmiştir. Uygulamada ana arayüz tmap olup, tmap üzerine açılacak yeni fxml dosyalarının yönetilmesi StageFactory sınıfında yapılmaktadır. Eklenen resmin click işleminin StageFactory içerisinde tetikleyeceği LogScreen ekranı için, StageFactory içerisine ilgili fonksiyonlar ve işlemler eklenmiştir. LogScreen AnchorPane nesnesine arka plan rengi verilerek uygulama çalıştırıp ilk test işlemi gerçekleştirilmiştir. Kayıt ekranı ekranın istenilen konumunda açılmıştır.

Kayıt ekranının açılma görevi bitirildiğinden bir sonraki göreve geçilmiştir. Fxml tarafından tanımlanan JFXTreeTableView nesnesi controller sınıfında build edilecektir. TreeTabın build edilmesi için aşağıda verilen Pojo sınıfı tanımlanmıştır. Bu sınıfın tanımlanmasının nedeni TreeTab’ın kolonlarının oluşturulmasında kullanılma zorunluluğudur. İçerisinde sadece nitelikleri ve bu niteliklere ait get ve set metotları bulunmaktadır. TreeTab’ın kolonunun oluşturulması ise aşağıdaki resimde görülmektedir.



Resimde görülen, kolon nesnesi oluşturulurken verilen “Receiver Time” metni ilgilini kolonun adıdır. Oluşturulan ReportLogPOJO sınıfının nesneleri, oluşturulan kolon nesnesinin, setCellValueFactory işlevine verilmektedir.



|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

TreeTab’a veri eklemek için Call For Fire raporunu loglanmak için seçildi. Logların veri tabanına kayıt edilmesi gerektiğinden, kayıtlar için veri tabanından 2 adet tablo oluşturuldu. Projede veri tabanı işlemleri için spring kullanıldığından ilgili tabloların sınıfları projede oluşturuldu. Kayıt işlemleri için 2 tablo bulunduğundan 2 adet servis sınıfı yazılmaya başlandı. Servis sınıflarında kullanılması için, servislere ait Repository interfaceleri oluşturuldu. Servislerin Repository nesneleri üzerinden veri tabanı işlemlerini gerçekleştirmesi için gereken Factory sınıfları oluşturuldu. Loglanma işleminde gelen raporun tüm içeriği json formatında veri tabanına kayıt edilecektir. Bunun için Call For Fire raporunun save ve create fonksiyonlarında gelen rapor nesnesi loglanmak için alınacaktır.

TreeTab’a verilerin eklenmesi için TreeItem kullanılmaktadır. TreeItem javaFx ile gelmektedir. Verileri tutuş şekli ise ağaç yapısına benzetilebilir. TreeItem nesnesi oluşturulduğunda, kendisi kök düğüm olmaktadır. İçerisine kendisi için atamalar yapılırken, kendisinin altına yaprak düğümler eklenmektedir. Yaprak düğümlerin sayısında sınırlandırma bulunmamaktadır. Her yaprak düğüm kendisi için kök düğüm olup, kendi altlarında yaprak yapılarını barındırabilmektedir.

TreeTab için kullanılacak TreeItem nesnesi ve veri tabanı işlemleri için aşağıda resimde görülen sınıf tanımlanmıştır.



Tanımlanan sınıf singleton olup springe devredilmiştir. Sınıf içerisine veri tabanı işlemleri için oluşturulan servis sınıflarından birisi ve TreeTab için ReportLogPOJO nesneleri tutacak TreeItem tanımlanmıştır.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

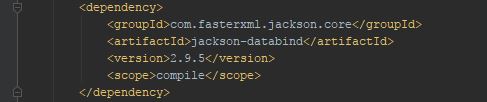
Log ekranına görüntülenecek log sayısında kısıtlama yapılacaktır. Kısıt sayısı 100 olarak belirlenmişir yani log ekranında sadece alınan ilk 100 rapor kayıtları gösterilecektir. TreeTab ReportLoggerHelper sınıfında bulunan TreeItem niteliğine bağlı olduğundan dolayı, TreeItem niteliğine eklenen veriler her zaman 0. İndexe eklenmiştir. Ekleme yapıldıktan sonra TreeItem’ın boyut kontrolü yapılmıştır. Boyutu 100’ü aştığı anda, 101. Nod silinmiştir. Bu ayar configuration tablosuna eklenerek, ayarın kontrolü kullanıcıya bırakılmıştır.

Call For Fire raporunun save ve create fonksiyonlarında loglama işlemi yapılacağından dolayı, log işleminin tekil olması için rapor sınıflarının parent sınıfı olan ReportService sınıfına reportLogger adında fonksiyon tanımlanmıştır. Call For Fire raporunun save ve create fonksiyonlarında parent sınıfında tanımlanan bu fonksiyon çağrılarak loglama işlemi başlatılacaktır.

ReportLogger sınıfı parametre olarak AbstractMesssage tipinde gelen veya gönderilen rapor, entity ve send\_receive bilgisini almaktadır. Bu fonksiyon Singleton olarak tasarlanan ReportLoggerHelper sınıfında oluşturulan addNewLog fonksiyonunu tetiklemektedir.

AddNewLog fonksiyonu gelen log bilgilerini hem veri tabanına hem de TreeItem listesine eklemektedir. Ekleme işlemlerinde raporun json formatına dönüştürülmesi gerektiğinden gerekli araştırmalar yapılarak aşağı resmi verilen modül maven ile projeye eklenerek gerekli işlemler yapılmıştır.

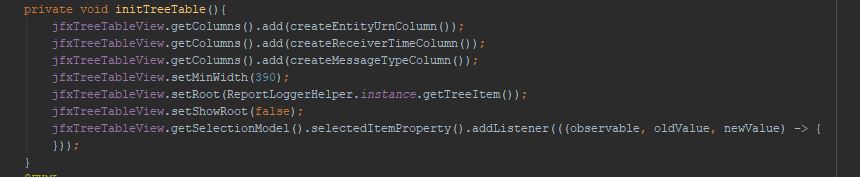
Veri tabanına ekleme işleminde hata ile karşılaşıldı. İlgili sütünün veri türü json, projede oluşturulan tablo sınıfında ise string olarak verilmişti. Veri tabanındaki alan json yerine varchar(3000) olarak değiştirilmiştir.



Loglama işleminin testi için kendi bilgisayarımda deneme yapıldı. Kendime entity göndermek için uygulama çalıştırılarak haritaya bir entity eklendi. Uygulama kapatılıp veri tabanında çeşitli değişiklikler yapılarak, eklenen entitynin uygulama için farklı bir entity olarak algılanması sağlandı. İlgili entitye Call For Fire raporu açılarak haritaya Call For Fire entityleri eklenerek veri tabanı ve ReportLoggerHelper sınıfın içerisinde bulunan TreeItem listesi kontrol edilmiştir. Yapılan kontrollerde bir sorunla karşılaşılmamıştır. Kayıt işlerinde bir sorunla karşılaşılmayınca TreeTab’ın oluşturulmasına devam edildi. Bir sonraki sayfada TreeTab’ın oluşturulması görülmektedir.

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |



|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

TreeTab oluşturulup uygulama çalıştırılarak test yapılmıştır. Daha önceden veri tabanına eklenen kayıtlar, başarıyla uygulama açılışında veri tabanından çekilmiştir. Log ekranı açıldığında önceden eklenen kayıtlar görüntülenmekte ve o an eklenen veya gelen rapor kayıtlarıda log ekranına düşmektedir.

TreeTab’ın görselinin düzenlenmesi için araştırmalara başlandı. Yapılan araştırmalarda aşağıda bir kısmı verilen css dokümanına erişilmiştir.

|  |  |
| --- | --- |
| .tree-table-view{  -fx-tree-table-color: rgba(0, 150, 0, 0.8);  -fx-tree-table-rippler-color : rgba(255, 0, 0, 0.9);}  .tree-table-view:focused .tree-table-row-cell:selected {  -fx-background-color: -fx-tree-table-color ;  -fx-table-cell-border-color: -fx-tree-table-color;}  .tree-table-view:focused .tree-table-row-cell:selected .tree-table-cell{ -fx-text-fill: white;}  .tree-table-view .jfx-rippler {  -fx-rippler-fill: -fx-tree-table-rippler-color;}  .tree-table-view .column-header ,  .tree-table-view .column-header-background,  .tree-table-view .column-header-background .filler{  -fx-background-color:TRANSPARENT;}  .tree-table-view .column-header{  -fx-border-width : 0 1 0 1;  -fx-border-color: #F3F3F3;} | .tree-table-view .column-header .label {  -fx-text-fill : #949494;  -fx-padding : 16 0 16 0 ;}  .tree-table-view .column-header .arrow, .tree-table-view .column-header .sort-order-dot{  -fx-background-color : #949494;}  .tree-table-view .tree-table-cell{  -fx-border-width:0 0 0 0;  -fx-padding : 16 0 16 0 ;}  .tree-table-view:focused {  -fx-background-color: -fx-tree-table-color,-fx-box-border,-fx-control-inner-background;  -fx-background-insets: -1.4, 0, 1;  -fx-background-radius: 1.4, 0, 0;  -fx-padding: 1; /\* 0.083333em; \*/}  .tree-table-row-cell .jfx-text-field{  -fx-focus-color: rgba(240,40,40); } |

Erişilen css dökümanı intranet ortamında TreeTab’ın base css’i ile birlikte incelenerek TreeTab’ın görsel tasarımı tamamlanıp çalışma haftası bitirildi.

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

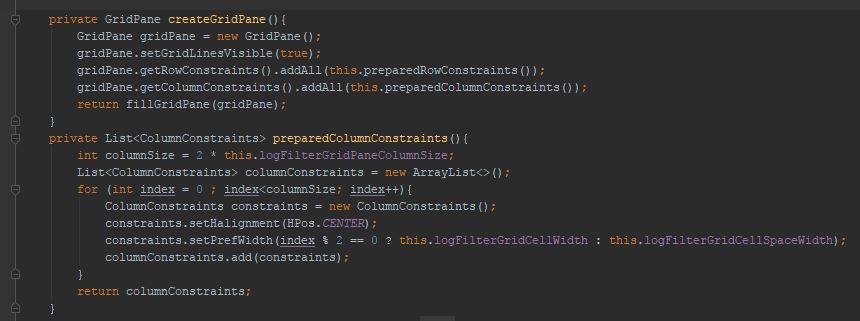
|  |
| --- |
| **YAPILAN İŞ: Filtre Ekranının Tasarlanması** |

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

Kayıt ekranının tasarımının bitilmesinden sonra, kayıtlar için filtrelerin yapılmasını sağlayan filtre ekranının tasarımına başlandı. İlk olarak filtre ekranının tetiklenmesinin nereden yapılacağı ve ekranın neresinde açılacağı konusunda ekipçe tartışıldı. TreeTab listesinin en son kolonun headeri içerisine sağ tarafta hizalanmış button üzerinden yapılmasına karar verildi. Bunun için ilgili kolonun headeri tasarlanmaya başladı. Headerin graphic niteliğine, oluşturulan HBox verilmiştir. HBox içerisine ise, headerin başlığının yazıldığı bir label ve button koyularak gerekli hizalamalar yapılmıştır.

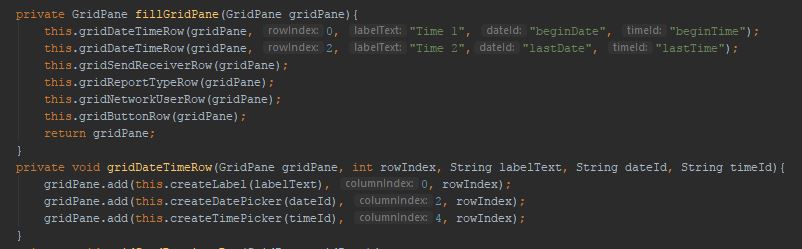
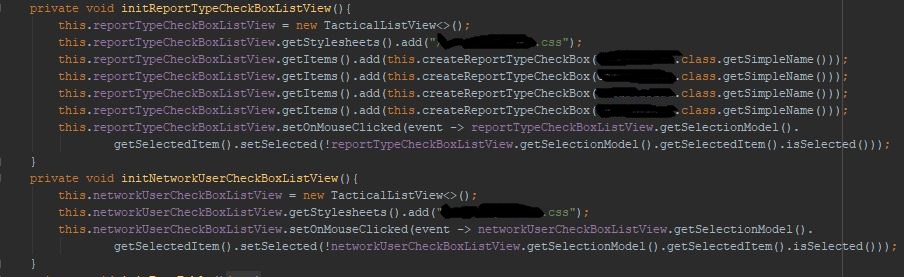
Filtre ekranının ise, log ekranının olduğu yerde açılmasına karar verilmişti. Filtre ekranı daha önceden oluşturulan TacticalDialog ile oluşturulup görüntülenecektir. Oluşturulan TacticalDialog contentine ise LogScreenController sınıfında oluşturulacak GridPane verilecektir. TacticalDialog sınıfı incelendiğinde gözden kaçırılan bir hatanın olduğu anlaşıldı. TacticalDialog sınıfı JFXDialog sınıfını extend etmektedir. İçerisinde projeye özel bazı tanımlamalar ve tasarımlar yapılmıştır. JFXDialog sınıfınta content niteliği ve bu niteliğe ait get set metotları bulunmaktadır. TacticalDialog sınıfında projeye özel content tasarlanmış ve bu content nitelik olarak TacticalDialoga verilmiştir ve base sınıfta bulunan getContent methodu ezilmiştir. SetContent methodu ise JFXDialog sıfına, yani parente bağlı kalmıştır. Oluşturacağım TacticalDialog nesnesinin contentine GridPane setlediğimde, get methodu ile çekmek istersem, fonksiyonun bana döneceği değer TacticalDialog içerisinde bulunan ve özel olarak tasarlanmış Dialog niteliği dönecektir. Dolasıyla GridPane üzerinde yapılan setlemelere erişemeyeceğim başka bir deyişle GridPane nesnesi kaybolacaktır. Bu hatanın düzeltilmesi için TacticalDialog sınıfında, parent sınıfa ait setContent methodu ezilerek gerekli işlemler yapılmıştır.

GridPane tasarımına başlanmıştır. GridPanenin esnek tasarlanması için solid’ e uygun olarak oluşturma aşamaları fonksiyonlara bölünmüştür. Tasarlanan GridPanede 16 row ve 6 columndan oluşmaktadır. 8 row ve 3 column boşluk için, 8 row ve 3 column ise tanımlanacak fxml bileşenleri için ayrılmıştır. GridPanenin tasarlanışının benzer örnekleri bir sonraki sayfadaki resimlerde görülmektedir.



|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |



GridPane yapısal ve görsel tasarımı bitirildiğinde LogScreenController sınıfındaki metot sayısı 44 olup toplam 600 satır koda ulaşılmıştır. CreateGridPane fonksiyonunda GridLines özelliğine true verilerek Gridin satır ve sütunlarının görülnülmesi sağlanmıştır. Bunun yapılmasındaki amaç, satır ve sütunların boyutlarının tam olarak ayarlanmasıdır.

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

Filtre ekranının testi için uygulama çalıştırıldı ve 2 sorunla karşılaşıldı. Filtre ekranı açıldığında log ekranı gizlenerek, log ekranının olduğu yerde filtre ekranı açıldı. İlk sorun eklenen ListViewlerin scroll özelliklerinin işlememesiydi. Yapılan araştırmalar sonunda hatanın nedeni anlaşıldı. Filtre ekranı ana stagede açılmıştır. Bu yüzden açılan dialog, log ekranının arkasında açılmıştır. Dialog ekranı ezildiğinden scroll işlemleri engellenmiştir. Dialogun log ekranının üstünde açmak için, dialogu log ekranına ait stagede açılması gerekmektedir. Bu durum tercih edilmemiştir. Sebebi ise, log ekranı ile filtre ekranlarının boyutlarının farklı olmasıdır. Bunun yerine, filtre ekranı ekranın tam ortasında açılmasına karar verildi. Bunu için gerekli hesaplamalar yapılarak, dinamik yerleşim için gerekli kodlar ve fonksiyonlar LogScreenController sınıfına eklenmiştir.

Ekrana eklenen TacticalCheckBox’lara yeni css’lerin özelliklerinin eklenmesi için styleSheet niteliğine yeni css dosyası eklenmiştir. Eklenen css dosyası Filtre ekranının görsel tasarımı için hazırlanmıştır. TacticalListView’lerin tasarımlarınında değiştirilmesi için css dosyasına yeni özellikler eklenip styleSheet niteliklerine css dosyası eklenmiştir. Bu durumda TacticalCheckBox’lar eklenen css dosyası ezilmiş tasarım bozulmuştur ve karşılaşılan 2. Problemde budur. Ezilmenin neyden kaynaklandığı bilinmemektedir. Ezilmenin düzeltilmesi için TacticalCheckBox’ların styleSheet nitelikleri temizlenerek tekrardan css dosyaları eklendiğinde sorun çözülmüştür.

Filtre ekranında yapılan seçimler configuration ayarlarına dâhil olmaktadır. Kullanıcının berlirlediği log filtre ayarlarını kaydettiğinde log ekranı yeni filtre ayarlarına göre doldurulmalıdır. Bunun için veri tabanından girilen filtreye uygun kayıtlar çekilecektir. Filtre ayarları configuration ayarlarına dahil olmaktadır. Uygulama tekrar açıldığında daha önceden kaydedilen filtre ayarlarına göre veri tabanından önceden kayıtlanmış veriler çekilecek ve uygulama çalışırken gelen rapolar ise kaydedilen filtre ekranına göre log ekranına düşecektir. Bunun için gerekli çalışmalar yapılmıştır. LogFilter sınıfı oluşturulup girilen log filtre ayarları bu sınıfta tutulmaktadır. Bu sınıfa erişim ise ReportLoggerHelper sınıfı üzerinden olmaktadır. Kullanıcı filtre ayarlarını değiştirdiğinde ReportLoggerHelper içerisinde bulunan LogFilter nesnesi json’a dönüştürülerek String veri tipinde configuration tablosuna yazılmaktadır. Uygulamanın açılışında configuration tablosundan filtre datası çekilerek json’dan LogFilter nesnesine dönüştürülmektedir. Belirlenen filtreye uygun kayıtların çekilmesi için aşağıdaki Spring CrudRepository sorgusu yazılmıştır.

List<ReportLog>findTop100ByUrnInAndReporttypeInAndSendreceiveInAndProcessTimeLessThanEqualAndProcessTimeGreaterThanEqualOrderByProcessTimeAsc(.......)

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

Filtre ayarları için, varsayılan mod oluşturulmuştur. Bu mod, raporların tarih filtrelerini uygulamanın açıldığı günden 1 gün öncesi ve 1 hafta sonrasına ayarlanır. Kullanıcı filtresini ise, sadece kendisinin gönderdikleridir ve tüm raporlar açıktır. Filtrenin varsayılan ayaları, filtre nesnesi oluşturulduğu anda kendi içerisine eklenen fonksiyonlar ile yapılmaktadır. ReportLoggerHelper sınıfı spring ile nesnesi oluşturulduğunda, sınıfın içerisinde bulunan bazı fonksiyonlar tetiklenmektedir. Tetiklenen fonksiyonlardan birisi configuration tablosundan filtre ayarlarını çeker. Gelen data eğer boşsa filtre nesnesi varsayılan ayarlar ile kalır. Gelen data boş değil ise, data filtre nesnesine dönüştürülerek ReportLoggerHelper sınıfının içerisinde bulunan filtre nesnesi güncellenir. Filtre ekranı açıldığında ise filtre nesnesinde kayıtlı olan ayarlara göre filtre ekranı doldurulmaktadır.

Filtre ekranı için yapılan gereksinimlerden sonra ekipçe React JS çalışılmaya başlandı. İlk olarak React JS için ön araştırma yapılmıştır.

React halen en güncel ve en çok talep gören önyüz kütüphaneleri arasındadır. Son yıllarda masaüstü uygulamalar azınlıkta kalırken, web ve mobil tabanlı uygulamalar giderek daha öne çıkmaktadırlar. Mobil uygulamaların popülerliği artsa da, web uygulamaları mobile’den daha esnek ve az maliyetli geliştirme ortamları sağlanmaktadır.  Kurulum ve kullanım kolaylığı ve platform bağımsız çalışabilmeleri gibi nedenlerle web uygulamaları bir süre daha önemlerini koruyacaktır. MVC tasarım modelinde sadece View katmanı ile ilgilenir.

React, Facebook yazılım ekibinin 2011 yılında kendi kullanımı için geliştirdiği bir kütüphanedir. Facebook’un Instagram’ı satın almasından sonra Instagram ekibinin bu yapıyı beğenerek kullanmak istemesi sonucu kütüphane revize edilerek Instagram’da kullanılmıştır.  2013 yılında açık kaynak kodlu hale getirilerek yazılım dünyasının kullanımına sunulmuş.  Kendi sitesinde React şu şekilde tanımlanmaktadır: **“Kullanıcı Önyüzleri Geliştirmek İçin Bir JavaScript Kütüphanesi”** .

React, bileşen tabanlı geliştirmeye yönlendirir. Bileşenler bir kere yazılıp farklı projelerde tekrar kullanılabilirler. Kendi durumlarını (state) yönetirler. Veriler değiştikçe sayfada sadece gerekli bileşenler güncellenir; bu da performansı artırır. Her işi farklı bir bileşen yaptığı için karmaşık sistemleri yönetmek ve hata oluştuğunda tespit etmek kolaylaşır.  Ayrıca, bu bileşenler mevcut web sitelerine de entegre edilebilir ve böylece mevcut projelerimizi React’ a parça parça geçirebiliriz.

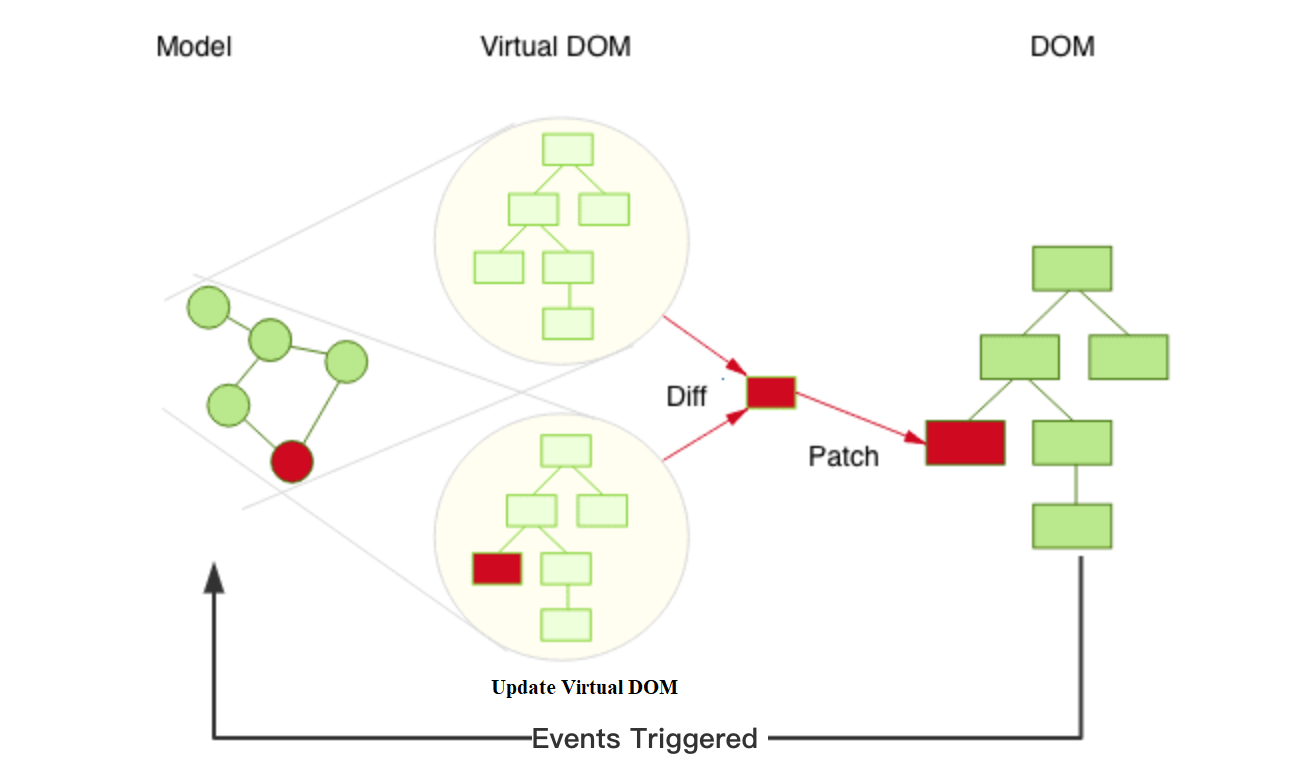
**React bu değişimleri performanslı bir şekilde yapabilmek için bellekte sanal bir DOM (Virtual Domain Object Model) tutmaktadır (VirtualDOM). Böylece yeni DOM’u sürekli eski DOM ile karşılaştırıp, farkları bulup, bileşenlerde sadece gerekli güncellemeleri yapmaktadır.**

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

DOM (Document Object Model) dediğimiz yapı; web sayfalarının bize görünmesini sağlayan, HTML elemanlarının hiyerarşik bir biçimde bir arada bulunması anlamına gelmektedir. Virtual DOM ise DOM yapısının anahtar => değer şeklinde memory’de bulunmuş haline denir. Yani DOM’un bir kopyasını almak gibi düşünebiliriz.

Virtual DOM için örnek vermek gerekirse, HTML web sayfanız üzerinde herhangi bir değişiklik yaptığımız anda, o değişikliğin ekranımıza yansıması için bütün DOM taranır ve ilgili değişiklikler oraya yansıtılır. Özetle bütün DOM baştan render edilmiş olur. İşin içine Virtual DOM girdiğinde ise herhangi bir güncelleme; ilk önce Virtual DOM üzerinde, yani bellek üzerinde yapılıyor. Daha sonra Virtual DOM’dan, gerçek DOM bileşenimize aktarılırken Virtual DOM ile gerçek DOM arasındaki farklılıklar kontrol ediliyor. Bu sırada React içerisinde bulunan farklılık algoritmaları devreye giriyor ve eğer herhangi bir farklılık yoksa render işlemi yapılmıyor. Farklılık var ise sadece ve sadece ilgili değişiklik yapılmış olan dallardaki farklılıklar render ediliyor. Aşağıdaki resimde Virtual Dom işlemi görülmektedir.



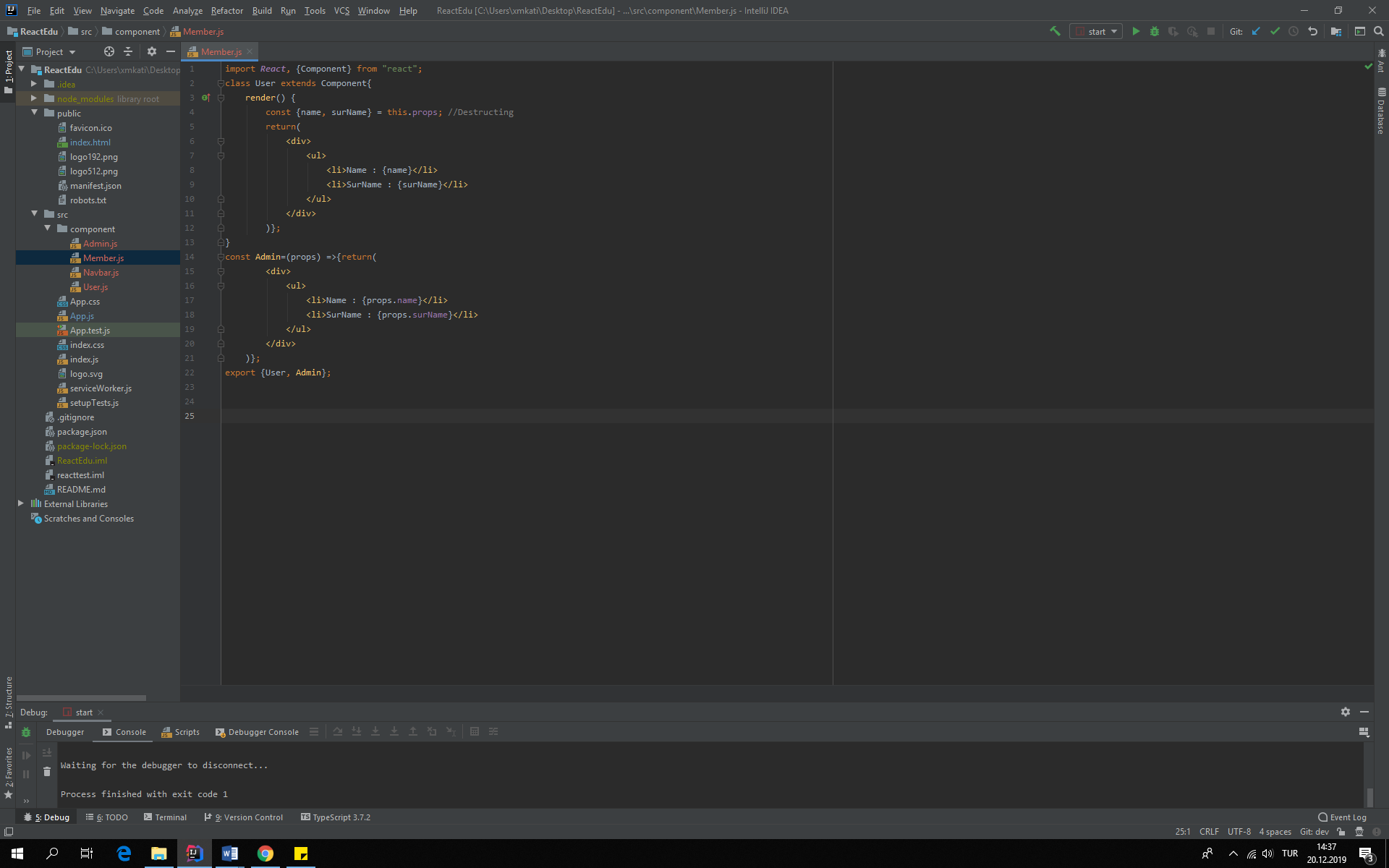
Virtual dom'a yeni bir güncelleme geldiğinde virtual dom kendisinin kopyasını oluşturur (Güncellenmeden önce oluşur.) Güncelleme işlemi bitirildiğinde eski kopya ile karşılaştırma algoritmaları ile karşılaştırma yapılarak nelerin güncellendiği bulunur. Bulunan değişiklikler reel doma yansıtılır.

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

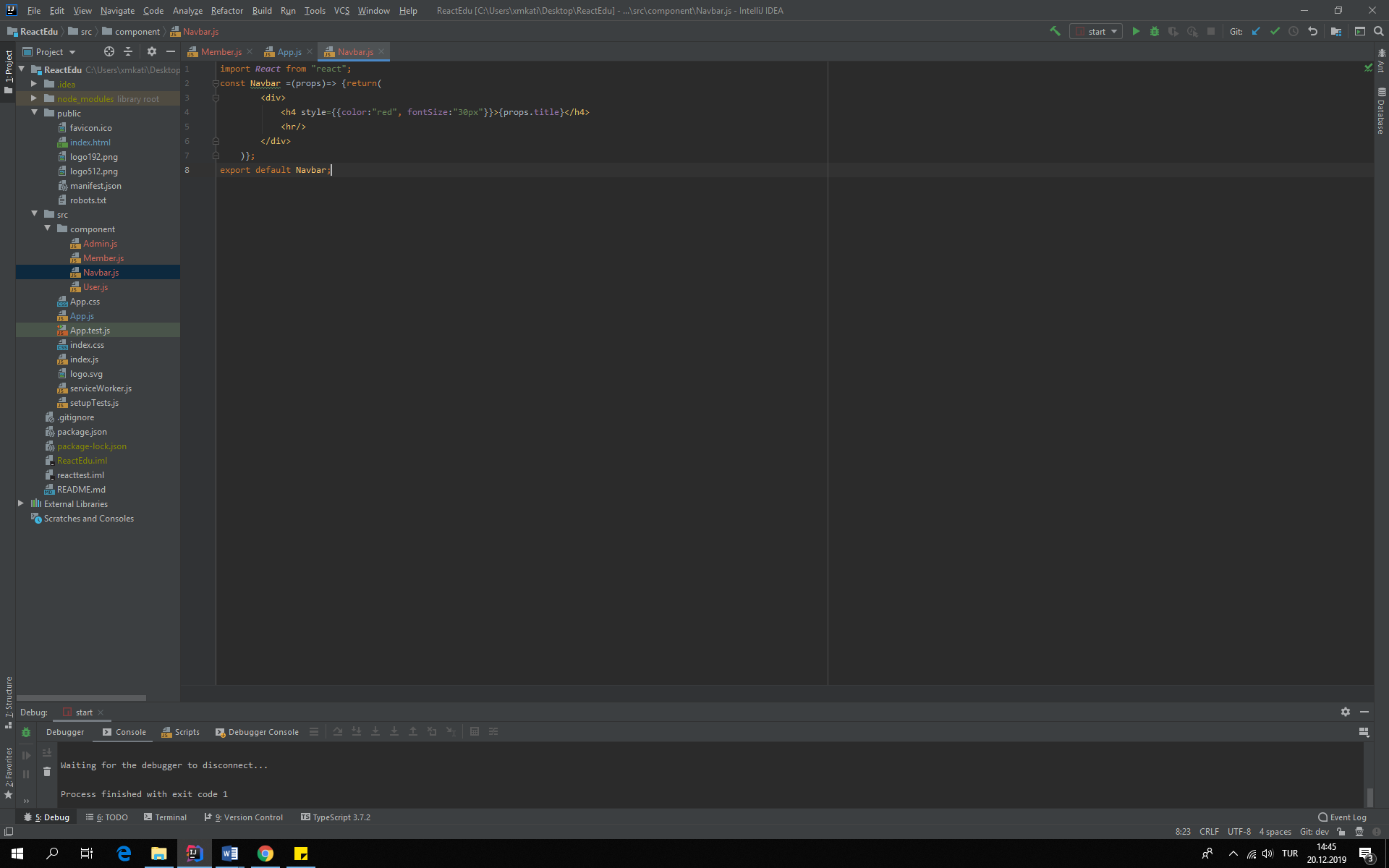
|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

React component tabanlıdır. Component yapıları sınıfa çevrilmektedir. Sınıfa çevrilen componentler içerisinde bulunan verileri state veya promp olarak gönderilmektedir. Bir defa yazılan component sınıfları her tarafta kullanılabilmektedir. Componentler birbirleri arasında properties yani props gönderebilmektedir. Component olarak hazırlanan sayfalarda, component bazında render yapılır. Jsnin 2015'den sonra gelen ES6 standartlarını kullanmaktadır. ES6 ile class yapıları gelmiştir. JSX formatı kullanmaktadır. React ile render yapılırken sadece bir tane element dönmektedir.

React JS ile çalışmalar yapılacağından gerekli kurumlar yapıldı. IntelliJ Idea ortamında ilk örnek yapılmıştır. Yapılan ilk örnek aşağıda görülmektedir.

Verilen resimde görüldüğü gibi 2 tip component tanımlanmıştır. User component’i class olarak tanımlandığı için react modülünde bulunan Component bileşenini extend etmektedir. Admin ise fonksiyon türünde oluşturulan component’ir. Resimde görülen props objesi ise, tasarlanan componentin başka componentler tarafından kullanıldığında, gönderilen parametrelere erişmek için kullanılır.

Tanımlanan başka componentler aşağıda verilmiştir.

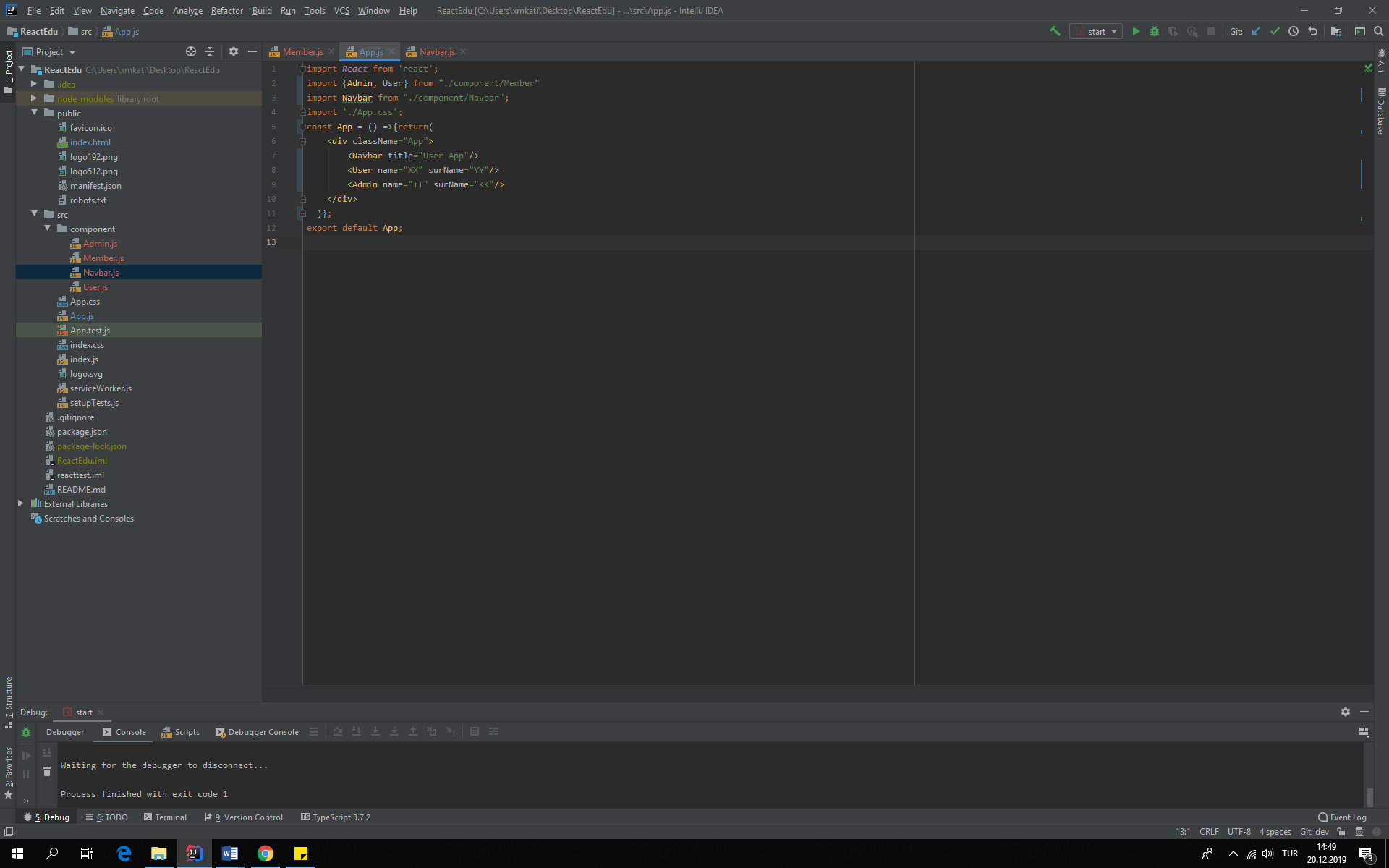


|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

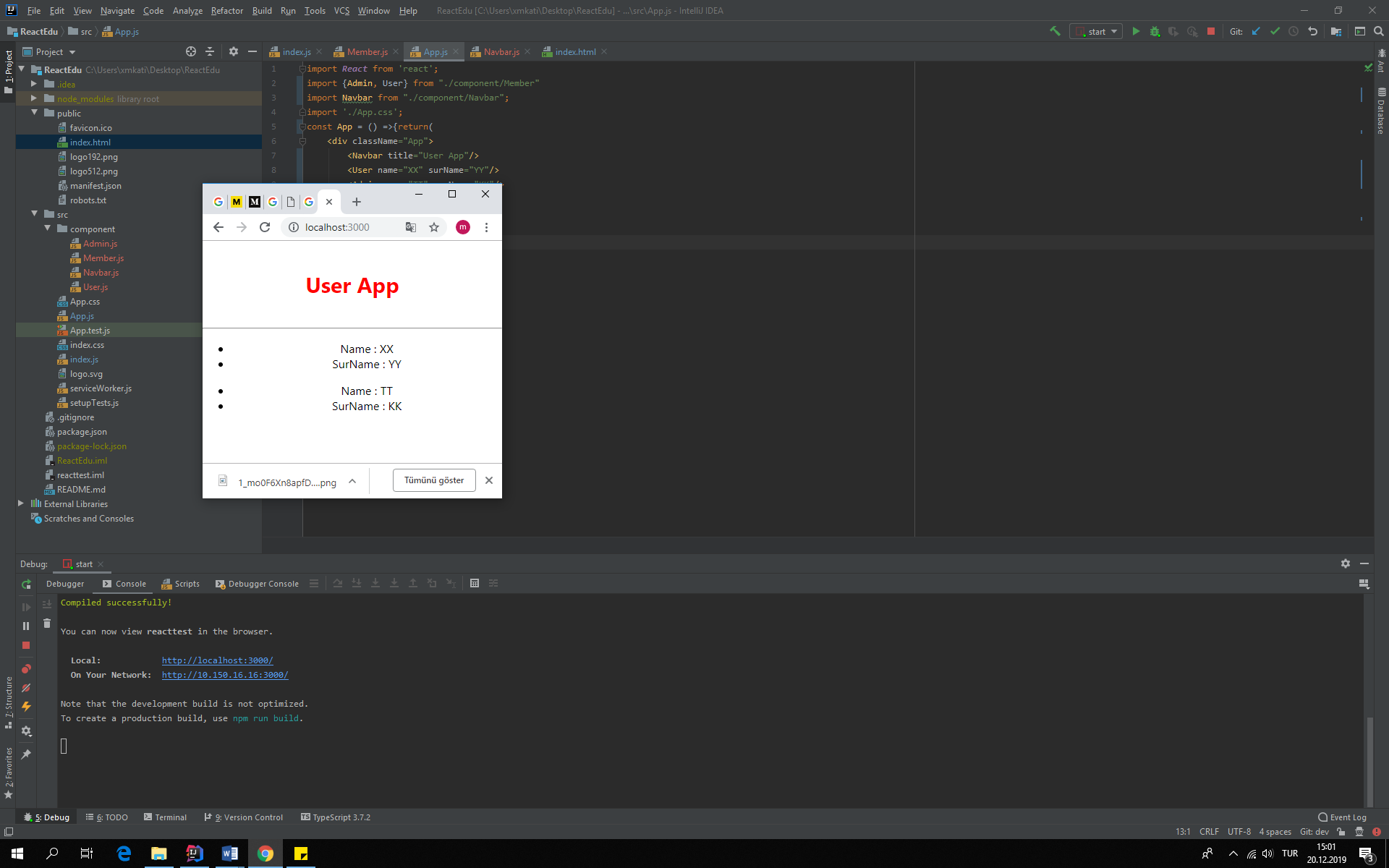
Yanda verilen resimde App componenti görülmektedir. Tanımlanan Admin, User ve Navbar componentleri kullanılmak için import edilmiştir. Yaratılan componentler Html tagları gibi kullanılmaktadır. Gönderilecek parametreler ise resimde görüldüğü gibi taglara nitelik olarak verilmektedir.

ReactDOM.***render***(<App />, ***document***.getElementById('root')); => Index.JS

<div id="root"></div> => Index.html



Yukarıda verilen JS kodu ile oluşturulan App componenti, Index.Js dosyasında, Index.html dosyasında tanımlanan div içerisine render edilecektir. Uygulama çalıştırıldığında aşağıda verilen çıktı görüntülenecektir.



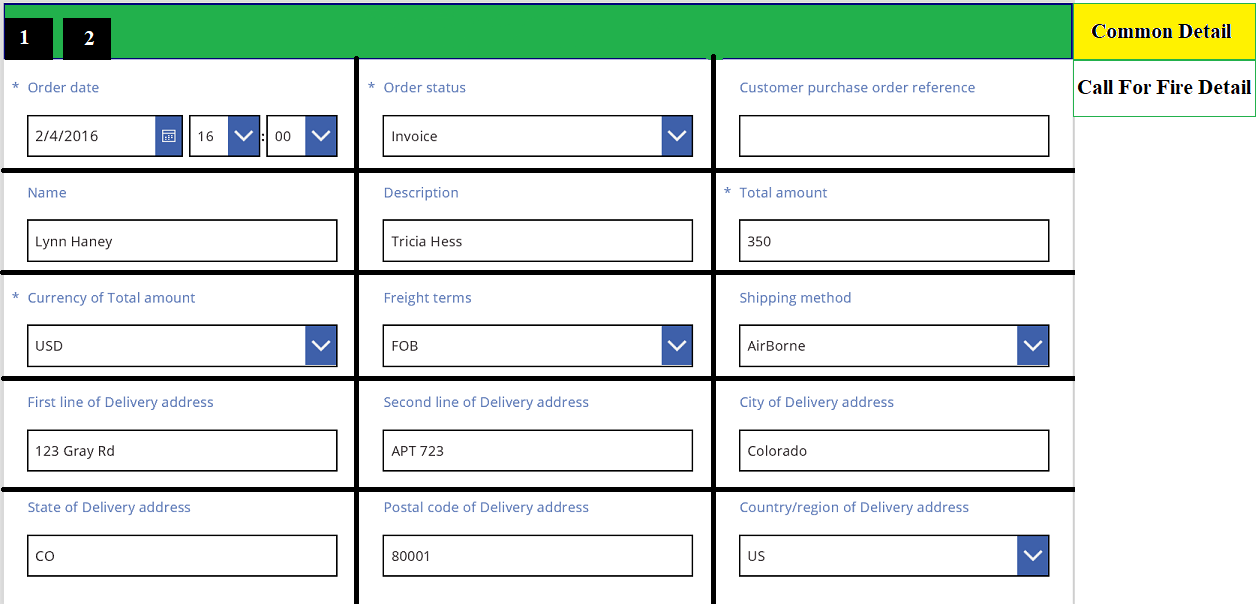
|  |  |  |
| --- | --- | --- |
| **DENETÇİ** | | **İŞYERİ YETKİLİSİ** |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |
| --- |
| **YAPILAN İŞ: Object Detay Pane Tasarımı** |

Projedeki en karmaşık sınıflar arasında olan ObjectDetailPaneController sınıfının iş yükünü azaltmak adına sınıfın yeniden düzenlenerek yazılmasına karar verildi.

ObjectDetailPaneController, entitylerin detaylarını göstermek için ekranın üstünden açılan bir formdur. Sınıfın satır sayısı 1026 olup, fxml dosyasının satır sayısı ise 1350’dir. Bu formun görüntüsünün paylaşımı yasak olduğundan internet ortamında bulunan örnek bir form üzerinden anlatılacaktır. Örnek form aşağıda görülmektedir.



ObjectDetail sınıfından açılan toplam 8 adet detay formu vardır. Bunlar;

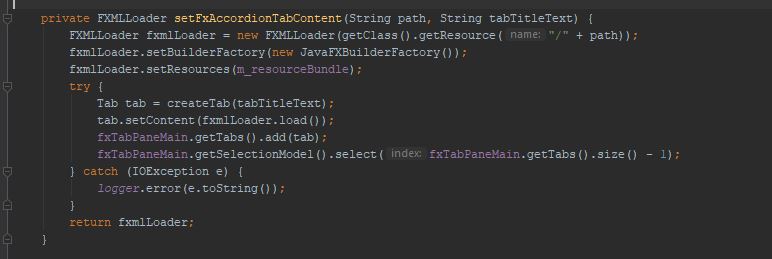
* CallForFireObjectDetail
* CbrnObjectDetail
* ClosedAirObjectDetail
* CommonObjectDetail
* EventObjectDetail
* IEDObjectDetail
* MedevacObjectDetail
* UnitObjectDetail

‘dır. Detai ekranlarının gözükmesinde JFXTabPane kullanılmaktadır. CommonDetail her entity tipi için görüntülenmektedir. Entity’lerin özelliklerine göre ise farklı formlarda görüntülenmektedir. Bun durumu özetlersek, harita üzerinde seçilen entity eğer CallForFire tipinde ise, detay ekranı açıltığında CommonDetail ve CallForFireDetail formu açılacaktır. Formlar yukarıda örnek resimdeki gibi grid yapısında tasarlanarak JFXTabPane’nin içerisinde bulunan tabların contentlerine verilmektedir.

CommonDetail içerisinde tekrardan JFXTabPane bulunmaktadır. Bahsedilen bu tablar verilen örnek resme bakıldığında, sağ tarafdaki kutular tab olarak geçmektedir. CommonDetail’a tıklandığında bu taba ait content gözükmektedir. Bu tabın içerisinde JFXTabPane bulunduğundan, bu tablar üstte gözükmektedir.

ObjectDetailPane fxml dosyasının 1350 satır olmasının sebebi açılan 8 detay formunun JFXTabPane içerisinde tab olarak kodlanmasıdır. 8 tab’a fx:id verilerek Controller sınıfında tanımlanmıştır. Bunların haricinde controller sınıfında 1 seçiçi tab tanımlanmıştır. Object formu açıldığında gelen entity tipine göre controller sınıfında tanımlanan seçici tab’a ilgili tab atanır. Gelen entity CallForFire ise, seçici tab CallForFire olmaktadır. Açılacak tab belirlendiğinde, JFXTabPane içindeki tüm tablar silinerek içerisine seçici tab ve CommonDetail tabı eklenir. Detay tablarının hepsinin sınıfı bulunmaktadır ve entity setleme işlemleri bu sınıflarda yapılmaktadır. Yani CommonDetail tabındaki değişiklikler CommonDetailController sınıfında yapılmaktadır. Common tabında bulunan fxml bileşeleri (listview, combobox vb.) parametre olarak ObjectDetailController sınıfından gönderilmektedir. Dolayısıyla ObjectDetailController sınıfında tüm detail tablarının controller sınıfları tanımlanmıştır.

İlk olarak ObjectDetail fxml dosyasında bulunan detail tabları aynı fxml dosyalarına yüklenip controller sınıflarının gerçekten controller olması sağlanmıştır. ObjectDetail fxml dosyasında sadece JFXTabPane bulunmaktadır. Gelen entity tipine göre, tablar oluşturularak contenxtlerine oluşturulan fxml doyaları load ile setlenmektedir. Aşağıda anlatılan işlem görülmektedir.



ObjectDetaildan çıkartılan tablar sayesinde, bu sınıfta tanımlanmış fxml bileşeni (listview, combobox vb.) kendi controller sınıflarına taşınmıştır. ObjectDetail sınıfının yükü hafifletilmiştir.

Detail tablarının controller sınıflarında, ortak metotlar bulunmaktadır. Bu metotlar;

* initUI
* setEntityFieldsWithFxValues
* Validate
* setFxData
* disableFields

‘dir. Her controller sınıfı entity üzerinde işlem yaptığından

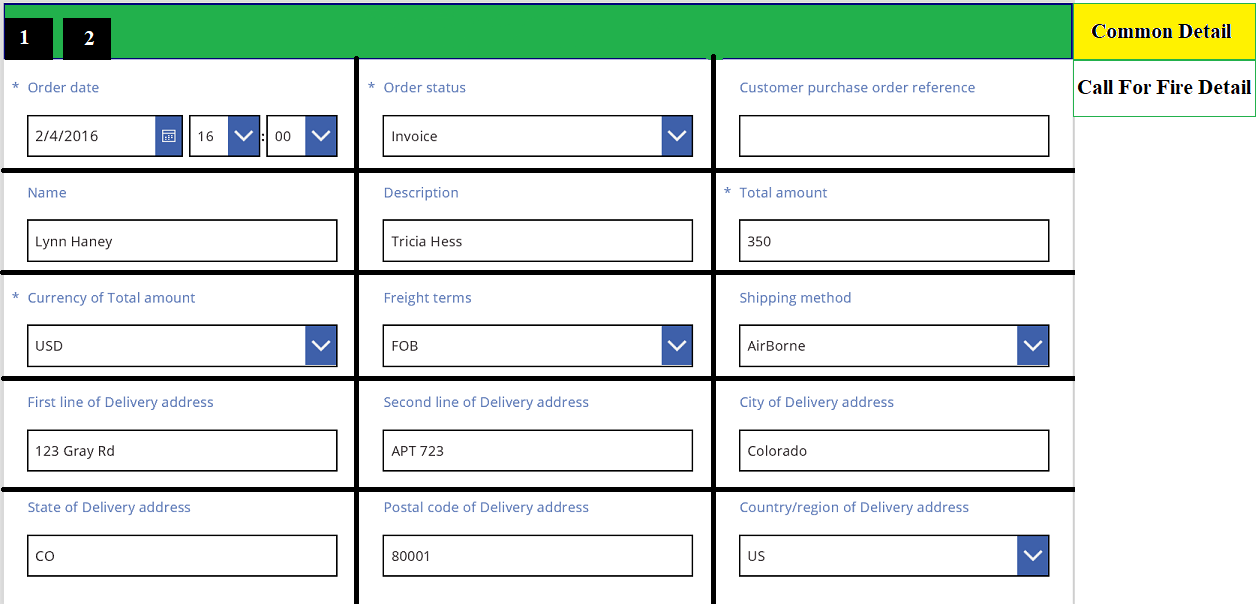
**Abstract bir sınıf** oluşturularak ortak field ve metotlar burada toplanarak, tabların controller sınıfları tekrardan yazılmıştır. Verebilirsen resim paylaş.

Bir entitynin detay ekranı açıldığında o entity’e ait tüm bilgiler görülüp değiştirilebilmektedir. Entitylerin alanlarının değiştirilmesinde bazı kısıtlar getirilmesi gerekmektedir. Harita üzerinde entityler oluşturuş durumuna göre 3 gruba ayrılmıştır. Bu gruplar Own, Own\_Create ve Networktur. Harita üzerinde seçilen entitynin grubu Own veya Own\_Create ise bilgilerin değiştirilmesine izin verilmektedir. Eğer seçilen entity’nin grubu Network ise, detail ekranı açıldığında tüm bilgiler görülmektedir fakat alanlar disable edilmiştir. Yani bilgilerde değişime izin verilmemektedir. Bu kısıtların uygulanması için entity grubunun adlarını tutacak bir enum oluşturuldu. Bu enum ile entity kontrol edilip grubu belirlenmiştir. Eğer entity Network’te ise disapleAllField fonksiyonu tetiklenerek tüm alanlar disable edilmiştir. Bu kontrol tüm controller sınıflarında yapılmıştır.

Tabların Controller sınıflarının ortak metotları **Abstract** sınıfta toplanınca ObjectDetail sınıfı tekrardan yazılmaya başlanmıştır. Tüm tabların controller nesneri silinerek sadece 2 adet nesne tanımlanmıştır. Bunlardan birisi CommonDetailController sınıfının referansı, diğeri ise tab’ların base abstarat sınıfının referansıdır.

Tüm DetailController ve controllere ait fxml dosyalarındaki düzenleme işlemleri tamamen bitirilip çalışma haftası bitirilmiştir.

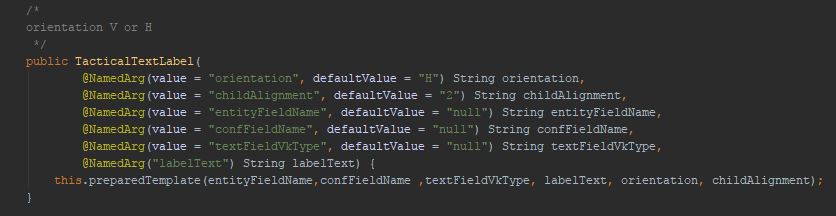
Geçen hafta bitirilen ObjectDetail düzenlemesi yapılacak olan çalışmanın ilk adımıdır. ObjectDetail panelinin düzenlenmesinin ikinci adımında projeye özgü componentler tasarlanacaktır.

ObjectDetail ekranında, raporların field alanları çok fazla oluğundan dolayı doldurulması gereken minimum 2 dolu form bulunmaktadır. Formlarda Listview, toggle button, checkbox, text field, combobox gibi bileşenler çokça bulunmaktadır. Text fieldleri değerlendirdiğimizde, açıklamasının olduğu bir label ve text field bulunmaktadır. Örneğin sağ tarafta verilen resimde text fieldin labeli üstündedir. Geliştirilecek özel component ile her bileşene otomatik label atanacaktır ve labelin bileşenin sağında, solunda, aşağıda veya yukarıda olduğu verilecek bir parametre ile otomatik ayarlanacaktır. Bunun getireceği en büyük performans ise, örneğin projenin Umman gibi ülkelere satılması durumunda labellerin zaman kaybetmeden sola alınmasıdır.

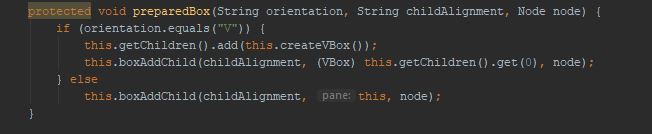
Tasarlanacak yeni componentin en önemli özelliği ise otomatik setlemedir. Detail panelinde yapılan değişiklikler onaylandıktan sonra tüm fxml bileşenleri gezilip, hatalı girişlerin olup olmadığı kontrol edilerek entityler güncellenmektedir. Bunu otomatik yapmak fxml bileşenleri, entity sınıfında etkileşimli niteliklere bağlanarak otomatik setleneme işlemi yapılacaktır.

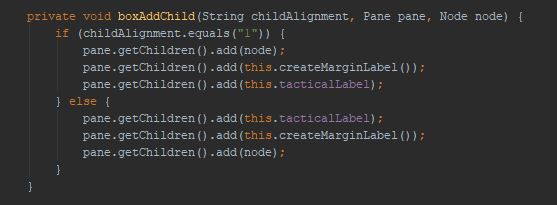
Anlatılanlar planlandığında yapılacak işlemlerin 2 adımda gerçeklenmesine karar verildi. İlk adım labellerin konumlandırılmasına ilişkin özel componentler, ikinci adım ise otomatik setlemedir.

Kodlamaya uzun planlamalar sonunda ilk adımdan başlandı. Özel component için özel label ve text filed sınıfları oluşturulup yazılımsal ve görsel tasarımları bitirilmiştir. Labellerin konumlandırılmasında HBox kullanılacaktır. Fxml tarafında tasarlanan component yazıldığında arka planda otomatik HBox içerisine label ve text field eklenerek konumlandırması yapılacaktır. Bunun için TacticalBoxParent sınıfı oluşturulup HBox’tan extend edilmiştir. Label ve yanına eklenecek fxml bişelenin konumlandırması burada yapılacaktır. TacticalTextLabel sınıfı oluşturulacak TacticalBoxParent sınıfından extend edilmiştir. Fxml tarafından eklenen bileşene nitelik gönderilmesi için kurucu metodu @NamedArg ile yazılmıştır. Aşağıda TacticalTextLabel sınıfının kurucu metodu görülmektedir.



Parametre olarak gönderilen VkType text fieldin numeric ya da alfabetic seçimidir. EntityFieldName ile entity’e bağlanılacaktır.

TacticalTextLabel sınıfında işlemler bitirilip, konumlandırma için base class olan TacticalBoxParent sınıfındaki ilgili metotlar tetiklenir. Base class’ın HBox’tan extend edildiği söylenmiştir. TacticalTextLabel kurucusuna gönderilen childAligment ile text field ve labelin konumu belirlenir. HBox yatayda hizalama yapmaktadır. Dikeyde hizalama yapmak için orientation V gönderilir. Bu durumda Hbox içerisine ilk olarak VBox yerleştirilir. VBox içerisine ise childAligment değerine göre yerleşim yapılır. Anlatılanlar aşağıda görülmektedir.

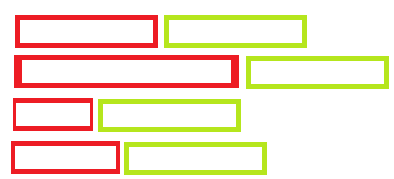


Verilen resimlerde fonksiyonların parametrelerinden birisine Node verilmiştir. Node fxml bileşenleri için base classlarındandır. Böylece textfield, combobox vb. fxml bileşenlerinin tasarlanmasında esneklik sağlanmıştır. BoxAddChild fonksiyonunda createMarginLabel fonksiyonundan belirlenen genişlikte label oluşturulup return edilmektedir. Buradaki amaç label ve yanına gelecek fxml bileşeni arasındaki mesafeyi belirlemektir. Hazırlanan text field componentinin artından toggle, checkbox, listview vb. bileşenler içinde componentler hazırlandı. Hazırlanan componentler bitirilince Detail fxmlleri tekrardan yazılmaya başlanmıştır. Aşağıda örnek tanımlamalar görülmektedir.

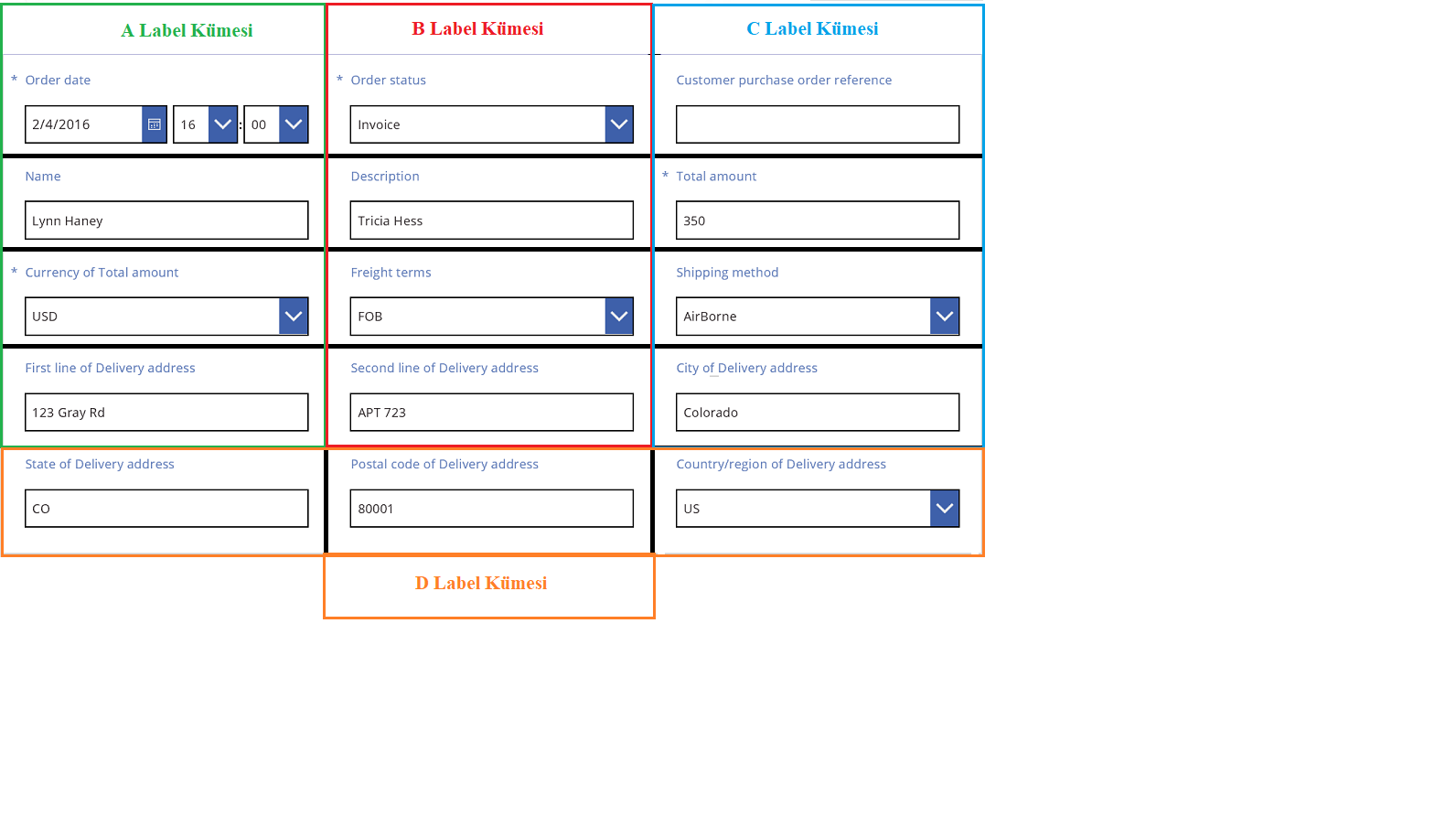


Bütün detail fxml dosyalarının yazımı bitirilerek çalışma haftası bitirildi.

Özel componentlerin tasarlanmasında son bir adım kalmıştır. Bu adımı bir örnek ile açıklayalım. Bir GridPane’nin bir kolonuna 4 adet TacticalLabelText eklensin. Hizalama ise H ve 1 olsun. Yani yatay hizalama yatayda, ilk label sonra text field eklenecektir. İlk bileşenin labelinde “xx”, ikinci bileşenin labelinde “xxxxxxxxxxxxxx” üçüncü bileşenin labelinde “x” ve son bileşenin labelinde “xxxx” yazıldığı varsayılsın. Labellerin textlerinin uzunlukları farklı olduğundan dolayı, kapladıkları genişlikler farklıdır. Bu durumda açılan formda bir düzensizlik olacaktır. Oluşan düzensizlik verilen resim ile örneklenmiştir. Kırmızı kutular labelleri, yeşiller ise textfieldlleri ifade etmektedir. Bir kolondaki tüm text field, combobox vs. niteliklerin aynı hizada başlaması için yanlarında bulunan labellerin genişliklerinin eşit olması gerekmektedir. Özel componentler için kalan son adım bu olup, labellerin dinamik boyutlarının ayarlanmasının sağlanması için araştırmalar başlamıştır. Yapılan araştırma ve planlamanın ardından bir yapı belirlenmiştir.



Planlanın yapıda labeller kümelenmektedir. Bu kümelemenin neye göre yapıldığı aşağıdaki resim üzerinden anlatılacaktır.



Bir formdaki labeller gruplanacaktır. Grup isimlerinde bir kısıt olamayıp istenilen ad verilebilmekte ve istenilen şekilde gruplar kurulabilmektedir. Verilen resimde 4 grup görülmektedir. Her grup, kendi içerisindeki en büyük genişlik değerini bulacak ve grup içerisindeki tüm labeller’e setlenecektir. Böylece form açıldığında, oluşacak düzensizlik engellenecektir.

Oluşturulan bu yapı bir noktada karışıklık yaratabilmektedir. ObjectDetail panelleri TabPane kullanmaktaydı. Örneğin CommonDetail panelinde 2 form bulunmaktadır ve sekme geçişleriyle üstten formlar değiştirilmektedir. Bu iki form CommonDetail fxml dosyasında bulunmaktadır. CallForFire entitysinin açıldığı varsayıldığında CommonDetail deki formlara ek formlarda eklenecektir. Buradaki labeller gruplandırılarak boyutlandırılacaktı. Daha sonradan debug yoluyla veya başka bir şekilde labellerin gruplarına bakılmak istenildiğinde, bir gurubun CommonDetail fxml dosyasında mı yoksa başka bir yerde olup olmadığının izlenmesinde sorun çıkabileceği düşünüldü. Bunun için label gruplarınıda gruplayan yeni bir grup daha eklendi. Böylece tasarlanan yapının son hali aşağıda görülmektedir.

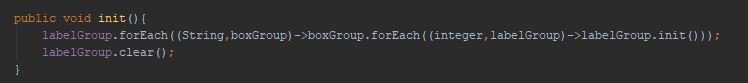
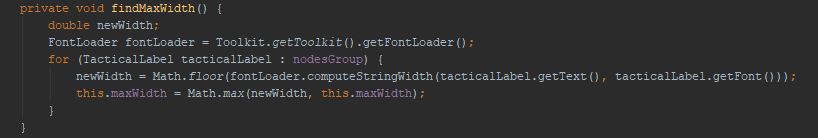
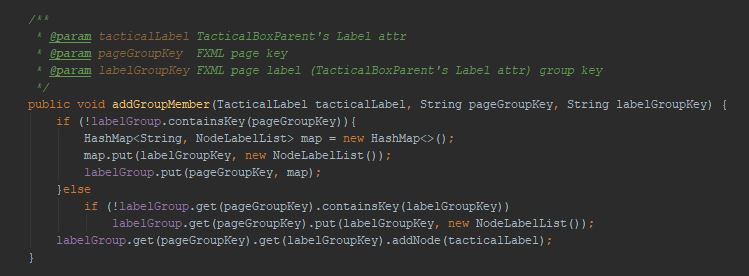


Verilen resmi bir tab/form olarak düşündüğümüzde, bu form 2 kümeye bölünmüştür. A ve D label kümesini içine alan Kapsayıcı Küme 2, B ve C label kümesini içine alan Kapsayıcı Küme 1. Böylece tasarlanan yapının verilecek adlar ile anlaşılması daha kolay olmuştur.

Label gruplarının sağlayacağı en iyi avantajlardan birisi, A label kümesinde text fieldlerin olduğu ve label açıklamalarının uzun olduğu varsayılsın. Bulunan genişlik değeri 190 olduğu varsayılırsa tüm labeller 190 genişliğinde olacaktır. D label kümesinde ise comboboxların olduğu ve label açıklamalarının çok kısa olduğu varsayılsın. Farklı grupta olacağından, label boyutlandırmaları kendi içlerine göre olacak ve yanlarına eklenecek comboboxların aralarındaki mesafenin görüntüyü bozacak kadar uzakta olması engellenecektir. Gruplandırmaları yan yana veya alt alta yapmak şart değil, çaprazlama, köşeler vb. şekillerde yapılabilip tasarımcının hayal dünyasına bırakılmıştır.

Bu gereksinimin gerçekleştirilmesi için oluşturulan componentlerin kurucu metotlarına @NamedArg(“pageGroupKey”) ve @NamedArg(“labelGroupKey”) nitelikleri eklendi.

Labellerin listesini tutmak için Singleton tasarım deseni kullanılarak ComponentCommon adında bir sınıf oluşturuldu. Bu sınıfta tutulacak labellerin listesini kapsüllemek için NodeLabelList sınıfı oluşturuldu. NodeLabelList labelleri tutmaktadır. İnit methodu ile bünyesinde bulunan en uzun textli labeli bulacak ve tüm labellerin genişliğini ayarlıyacaktır. ComponentCommon sınıfı ise label ve page gruplarını yönetmektedir. Sınıfların içleri doldurulup, düzenlenen component yapısına göre tüm fxml dosyaları düzenlenip çalıştırılmıştır. Proje çalıştırıldığında beklenilen sonuç elde edilememiştir. Nedeni ise, formunların gösterilmeden önce genişlik hesabının yapılmasıdır. Bu durumda labellerin içlerinde text olsada, daha görüntülenmediği için genişlikleri sıfır gelmektedir. Araştırmalar sonucunda bir text’in font bilgisine göre boyutunu hesaplayan FontLoader sınıfı kullanılmaya karar verildi. Labellerin genişliğinin hesaplandığı fonksiyonlar FontLoader sınıfı ile güncellenip proje çalıştırılmıştır. Yapılan testlerde bir sorunla karşılaşılmamıştır. Yapılan işlemlere ait bazı görseller aşağıda görülmektedir.



Oluşturulan yeni componentlerin kurucu metotlarına gönderilen field adlarıyla entitylere bağlanılacağı söylenmişti. Bunun için TacticalEntityTextField sınıfı oluşturulup, TacticalTextField sınıfından extend edilip EntityNode arayüzü implement edilmiştir. TacticalTextField sınıfı özel componentler için oluşturulmuştur. EntityNode arayüzü ise doldurulacaktır.

TacticalEntityTextField sınıfında 2 parametre bulunmaktadır. String veri türünde fieldMethodName adlı parametre ve Method veri türünde fieldSetMethod adlı parametredir. Yapılacak işlem, entity nesnesi reflection ile gezilerek fielde ait get ve set metotlarını bulmaktır. Bunun için entity Object sınıfına kadar taranacaktır. CallForFire tipindeki bir entity için base sınıfı olan Entity fieldlerine erişilmemektedir. Bahsedilen tarama işlemi bu yüzden yapılacaktır. Entity’nin bu sınıfa setlenmeden çekilmesi için Singleton tasarım deseni kullanılarak StageEntity adında bir sınıf oluşturuldu. DetailController sınıflarının base sınıfından entity fieldi alınarak StageEntity içerisinde oldEntity ve newEntity adında iki field tanımlanmıştır. Tüm detail işlemleri için entity kullanımları bu sınıf üzerinden yapılacağından detail sınıfları üzerinde değişiklik yapılması gereken yerler yeni yapıya göre yeniden yazılmıştır. Otomatik setleme yapılacağından, setlemeler newEntity üzerinden yapılacaktır. Setlemelerin iptal edilmesi durumunda geriye oldEntity döndürülecektir.

EntityNode arayüzünde tanımlanması gereken default fonsiyonlar tanımlanmıştır. İlk olarak ilgili fieldin get methodu ile değeri çekilerek TacticalEntityTextField textine setlenecektir. TacticalEntityTextField’e focusedProperty’sine listener eklenerek focus olduğunda veya unfocus olduğunda otomatik setleme işlemi yapılacaktır. Bunun için yazılan kod aşağıda görülmektedir.

TextField componentinin haricinde combobox, checkbox gibi componentlerin entity bağlantıları, componentlerin tasarımları, DetailController sınıflarının ve fxml dosyalarının yeniden düzenlenmiştir.

