|  |  |  |
| --- | --- | --- |
| 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.. |

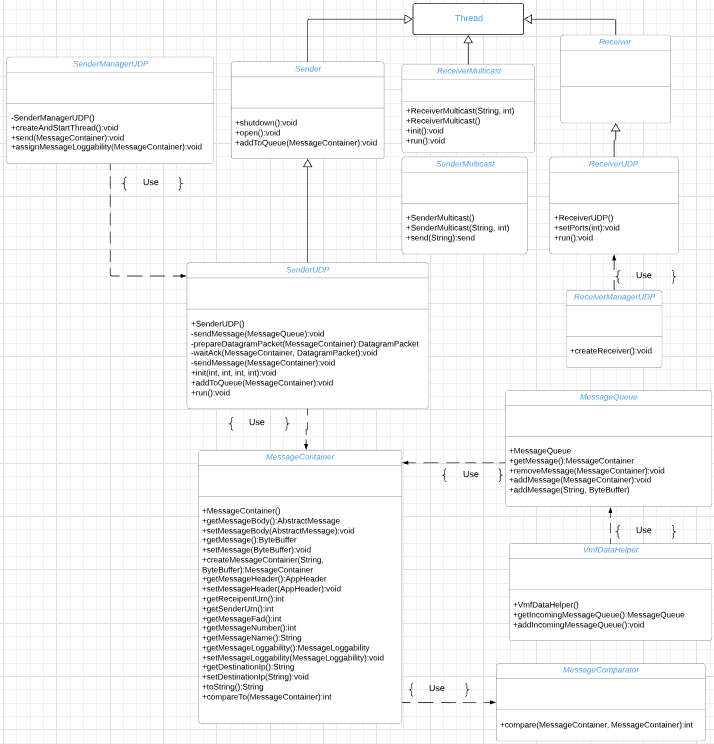
|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |



|  |  |  |
| --- | --- | --- |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |  |
| --- | --- | --- |
| 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 |  |

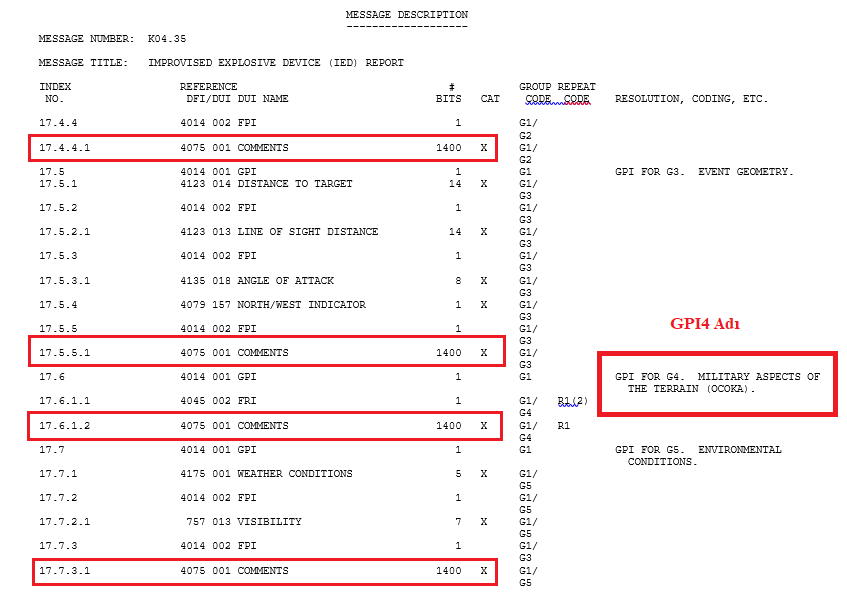


|  |  |
| --- | --- |
| Karadeniz_Teknik_Üniversitesi.gif | **T.C.**  **KARADENİZ TEKNİK ÜNİVERSİTESİ**  **OF TEKNOLOJİ FAKÜLTESİ**  **YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**  **İŞYERİ UYGULAMASI HAFTALIK RAPOR** |
| Tarih | / / 20.. |
| Sayfa No |  |

|  |
| --- |
| **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.



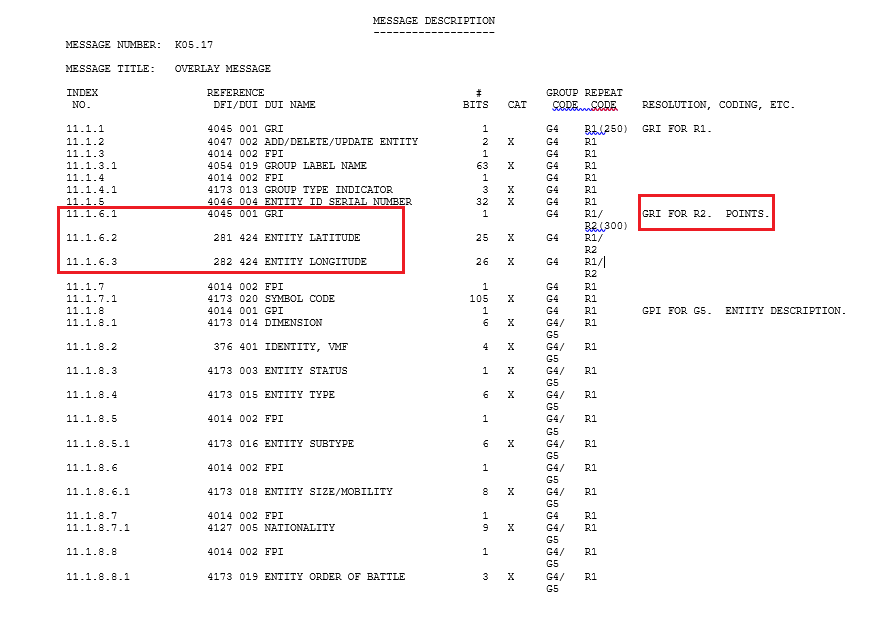
|  |  |  |
| --- | --- | --- |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

|  |  |
| --- | --- |
| 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.



|  |  |  |
| --- | --- | --- |
| 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.

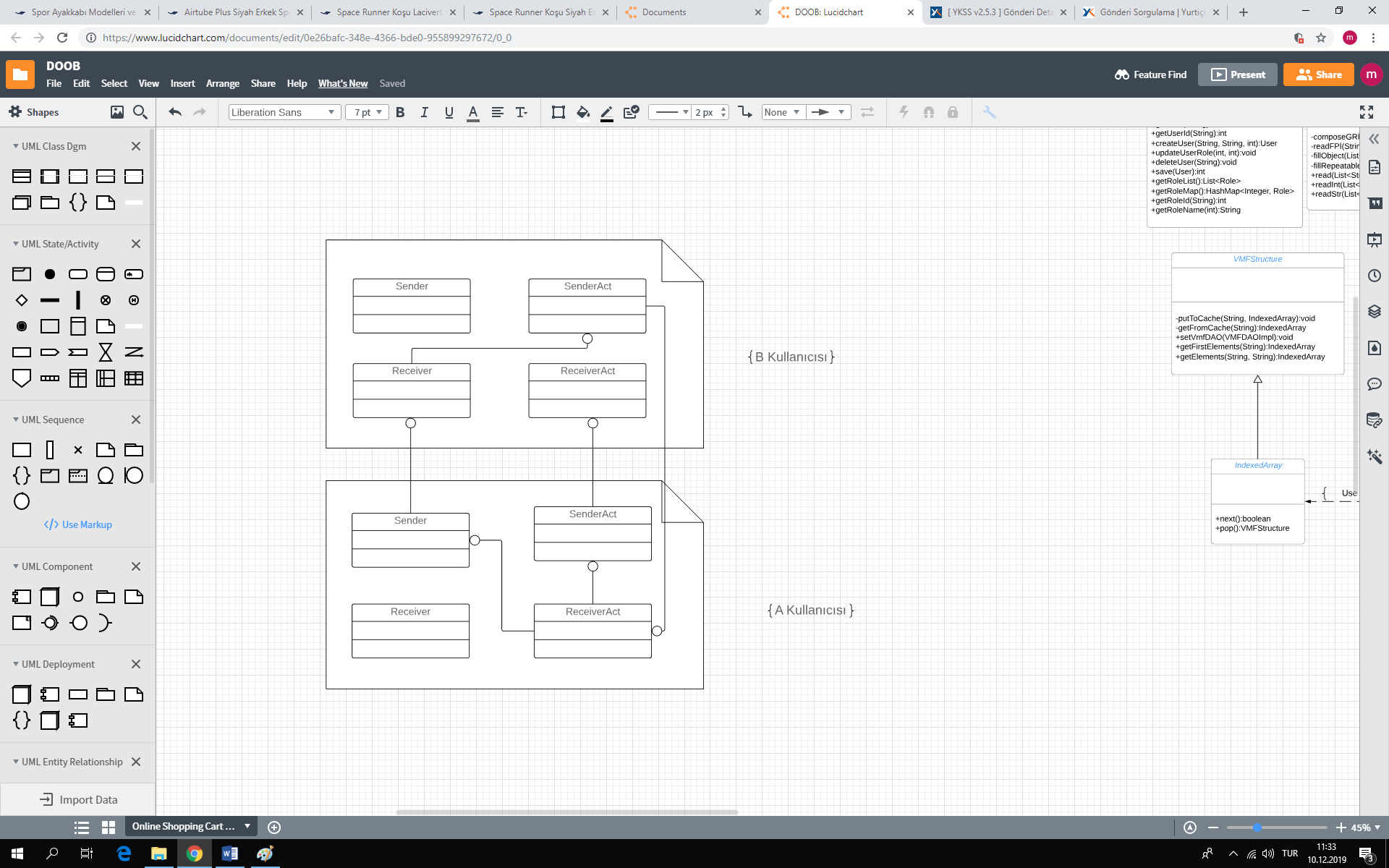
Overlay raporunda karşılaşılan sorun, Layere 5’den fazla entity eklendiğinde eklenen entitylerden sadece bazılarının gitmesidir. Gönderilen entitylerin alınma sayısında değişiklik olup olmadığının kontrolü için tekrardan test yapıldı. Yapılan testin sonucunda alıcı tarafa maksimum 4 entity iletildiği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

|  |  |  |
| --- | --- | --- |
| 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 |  |

Muzaffer bey ile daha önceden konuşulan UDP işlemleri ile ilgili tekrardan toplantı yapıldı. Yapılan toplantı neticesinde UDP işlemlerinin aşağıda resimde görülen yapıda – benzer olması kararlaştırıldı. Resimde A kullanıcısı bir raporu B kullanıcısına gönderme işleminde sınıfların bağımlılıklarını göstermektedir. Projede bu değişikliğe bu hafta yapılacak toplantı sonrasında başlanmasına karar verildi.



|  |  |  |
| --- | --- | --- |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

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.

|  |  |
| --- | --- |
| 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.

|  |  |  |
| --- | --- | --- |
| 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.

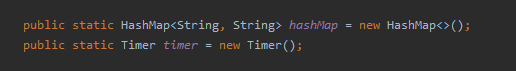
Soyut bir sınıf olan **TimerTask** **Runnable** arabirimi uygular ve dolayısı ile aslında bir [Java Thread](http://www.fatihkabakci.com/Makaleler-JAVA_THREAD_OLUSUMU) olarak çalışır. Bundan dolayı run() metodunu override ederek, gerçekleştirmek istediğiniz task kodunu, bu metoda yazılması gerekmektedir. 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.

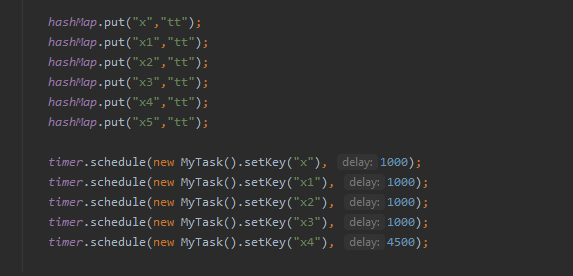
|  |  |  |
| --- | --- | --- |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

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

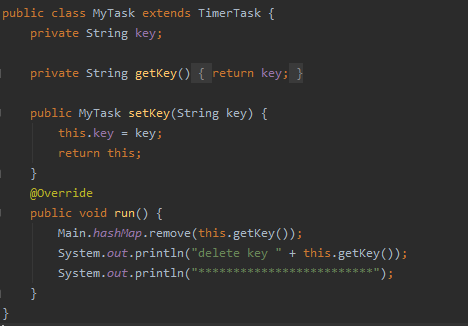
Timer yapısının nasıl kullanılacağı hakkında ön bir çalışma yapılmaya başlandı.



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.

|  |  |  |
| --- | --- | --- |
| 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 değiltirilecektir. Açıklanan 2 durumdan kopyalanma tercih edilmiştir.

Yapmak için üstlendiğim gereksinimin değiştirilmesine karar verildi. Üstlendiğim yeni gereksinim, raporların loglanmasıdır. Bunun için gelen ve gönderilen raporların loğları tutulacak ve kullanıcı log ekranını açtığında log dökümanına erişecektir. Listelenen loglar arasından seçilen bir log olursa, ekranda bir panel açılarak ilgili raporun içeriği kullanıcıya gösterilecektir. Loglar için filtre uygulanabilecektir; zamana veya zaman aralığına, kullanıcılara (kendisinin gönderdiği veya seçilen kişi/kişilerden gelenler), rapor tiplerine göre loglar 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.

|  |  |  |
| --- | --- | --- |
| 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.

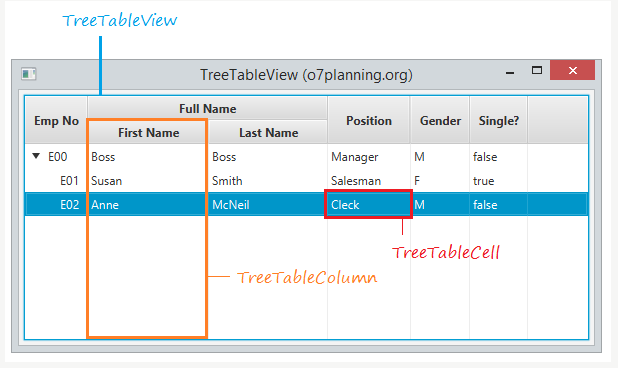
Yazılan bazı

|  |  |  |
| --- | --- | --- |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |

Story örnekleri;

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

Raporların loglanmasında esnek tasarım 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.

|  |  |  |
| --- | --- | --- |
| Onay (İmza & Kaşe) | | Onay (İmza & Kaşe) |
| / /20.. |  | / /20.. |