High Level Design of
Agile Express Project

Scrum Project Management System

OBSS 2022 Java Chellanger Program Project Design

09.08.2022

Selman Burak Karagöl

İçerik

1. Giriş

1.1. Neden High Level Design ?

Bu Üst Düzey Tasarım (HLD) Belgesinin amacı, kodlama için uygun bir modeli temsil etmek için mevcut proje açıklamasına gerekli ayrıntıları eklemektir. Bu belge aynı zamanda kodlamadan önce çelişkileri tespit etmeye yardımcı olmayı amaçlamaktadır ve modüllerin yüksek düzeyde nasıl etkileşime girdiğine dair bir başvuru kılavuzu olarak kullanılabilir.

1.2. Kapsam

HLD belgeleri, veritabanı mimarisi, uygulama mimarisi (katmanlar), uygulama akışı (Navigasyon) ve teknoloji mimarisi gibi sistemin yapısını sunar. HLD, sistem yöneticileri tarafından anlaşılması gereken teknik olmayan ve orta düzeyde teknik terimler kullanır.

1.3. Genel Bakış

Bu belgede anlatılacaklar:

- tüm tasarım yönlerini sunmak ve bunları ayrıntılı olarak tanımlamak
- uygulanmakta olan kullanıcı arayüzünü tanımlamak
- donanım ve yazılım arayüzlerini tanımlayın
- performans gereksinimlerini açıklamak
- tasarım özelliklerini ve projenin mimarisini içerir
- list and describe the non-functional attributes like:
  - security and reliability
  - maintainability
  - portability
  - reusability

2. Proje Detayları

2.1. Ürün Perspektifi

Agile Express, agile metodolojisi ile çalışan takımların kullanabileceği bir takip ve yönetim uygulamasıdır. Uygulamada 4 farklı kullanıcı rolü vardır. Bunlar yetki düzeylerine göre: ‘Admin’,’ProjectManager’,’TeamLeader’,’TeamMember’ şeklindedir. Bu kullanıcılar projedeki kısımları yetkilerine göre görür ve işlem yapabilirler. Uygulamada scrum tipinde çalışan takımların projolerini oluşturup yönetebilecekleri yapılar mevcuttur. Bu yapıların başında sprint oluşturma gelir. Bir sprint için 3 farklı ‘state’ mevcuttur. Bu stateler: ‘Aktif’,’Planning’ ve’ Completed’ şeklindedir. Bir sprinti aktif konumuna getirmek demek, bu sprintin o anda projede bulunan kullanıcılar tarafından üstünde çalışıldığı anlamına gelir. Bir sprint içinde tasklerin hangi konumda olduğunun takip edilmesi gerekir. Bunu ise proje oluşturulurken kullanıcının girdiği task statulerine göre belirleriz. Taskler bu statuler ve Backlog Task’leri olmak üzere 2 gruba ayrılırlar. Backlog Task’ler projenin herhangi bir sprinti ile bağlanmamış olan task grubuna denir. Takım Lideri veya daha kıdemli bir kullanıcı bu taskleri sprintlere ekler ve aktif sprint haline getirmesi durumunda koşu başlamış olur. Takım üyeleri ise bu taskleri alıp uğraşmakla sorumludur. Her takım üyesi kendi üzerinde bulunan Task için ne kadar çalıştığını ve ne yaptığını tutan bir çok Log oluşturabilir ve bu Log datalarını düzenleyip silebilirler. Bu sayede uçtan uca yetkilendirilmiş ve basit bir şekilde Scrum takibi yapılabilecek bir proje ortaya çıkmış olur.

2.2. Kullanılan Araçlar ve Teknik Seçimler

1. Lombok: Neredeyse tüm Class’larda bulunan Getter,Setter,Constructor gibi yapıları Lombok kullanarak Anottation ile belirtmek okunabilirliği arttırdığı için tercih edilmiştir.
2. Docker: Database’leri docker üzerinden çalıştırmak proje ile olan bağlantısını kesip dağıtık bir sisteme geçildiğinde de işleri kolaylaştıracaktır.Bunun yanı sıra hızlı bir şekilde açıp kapatabileceğimiz bir yapı oluşur. Fakat database’in verileri docker içinde tutmaması gerekir çünkü docker bir işletim sistemi proccessi gibi çalışır ve kapatıldığı durumda veriler silinir.
3. JPA: Jpa Repository, entity annotationı ile oluşturduğumuz entityler üzerinden database işlemleri yapmamızı sağlar.Bu bize düzgün bir database bağlantısı ve temiz bir kod sağlar.Aynı zamanda ilişkili tablolar üzerinde işlem yapmayı kolaylaştırır.

4. JWT: Jwt günümüzde CSR tabanlı web frameworkleri için en güvenli oturum yöntemleri arasındadır.
5. MongoDB: MongoDB döküman tabanlı bir veri tabanıdır. Projede kullanılma sebebi esnek çalışabilen ve kolay arama yapabildiğimiz bir yapı olmasıdır. Çok fazla ilişki bulunan durumlarda verinin güvenliğini ve consistency sağlamak için tercih edilmez fakat projede tablo sayısı az olduğu için bu durumları mongo tarafında sağlamak, ilişkisel veritabanına harcayacağımdan daha az zaman alacağı için MongoDB tercih edilmiştir.

6. ElasticSearch: Elastic Search database olarak değil daha çok search engine olarak kullanılması hedeflenmiştir. Burada arama yapmak istediğimiz dataları elastic search tarafına indexleriz ve bu datalar üzerinden Full Text Search yapabilme özelliğini kazanırız. Burada dikkat edilmesi gereken konu ES ve MongoDB tarafındaki dataların tutarlı olmasıdır.
7. Vue JS: Vue.js güncel Client Side Rendering tabanlı js. Frameworkleri arasında en esnek olanlarından biridir. Bu sayede hızlı bir şekilde proje geliştirebiliriz. Bunun yanında Vue3 için olmasa da Vue2 için oldukça fazla hazır component bulunmaktadır. Topluluk desteği 2022 yılında React.js kadar olmasa temel gereksinimleri kolaylıkla bulabileceğimiz kadar yeterlidir. Aynı zamanda Vue framework'un esnek yapısı işleri çözmenin birçok alternatif yapısını destekler. Bu sayede topluluk desteği daha az olsa bile React.js tarafında zorlanarak yaptığımız bazı işleri kolaylıkla çözebiliriz.
8. Vuex: Component tabanlı frameworklerin en büyük challenge'larından biri de datanın nasıl bir şekilde saklanacağıdır. Genel kullanım datayı componentler arasında transfer ederek kullanmaktır fakat bu durum component sayısı arttığında ve çok fazla ortaklaşan data olduğunda içinden çıkılamaz bir hal alır. Bu sorunu Vuex ile Store yapısını proje için tekil hale getirerek çözeriz. Oluşturduğumuz Global State ile tüm componentlerden erişilebilir bir Store yapısı kurmuş oluruz. Projede hem Vuex hem component tabanlı State yapısı ihtiyaca göre kullanılmıştır.
9. Vuetify: Vue.js dilinin mimarı Evan You tarafından desteklenen Vuetify bir tasarım kütüphanesidir. Projede hazır card yapıları ve form itemlar Vuetify üzerinden kullanılmıştır.
10. Proje Dosya Yapısı : Backend tarafında temel dosya yapıları Controller,Service,Repository,Documents,Domains şeklindedir. Burada Controller api isteklerinin karşılandığı kısımdır. Service katmanı bussiness logiclerin yazıldığı databaseden önce datalara eriştiğimiz mantık katmanıdır. Documents MongoDB ile ilişkilendirilecek dataların Map'lendiği ve nasıl tutulacağının belirlendiği classlardır. Repository, Document'lerin database tarafında işlemlerinin yapıldığı kısımdır. Domains, database datası ile frontende gidecek datanın birbirinden ayrılmasını sağlayan Class'larımızdır (Projede daha çok response handler gibi kullanılmıştır).

2.3. Rollere Göre Yetki dağılımı

Uygulamada 4 farklı rol bulunmaktadır, bunlar:

-Admin:

Uygulamada yapılabilecek tüm özellikleri yapabilen en yetkili kullanıcıdır. Fakat scrumlara veya projelere dahil edilemez. Yalnızca projeyi oluşturduğunda projede gözükecektir.

- Project Manager:

Bir projede görev alabilir. Uygulamada Proje oluşturabilir ve kendine ait olan projelerle alakalı tüm işlemleri yapabilir. Bu işlemler Sprint,Task,TaskLog,User işlemlerini de kapsar.

- Team Leader:

Bir Projede Tasklerle alakalı işlemleri yapabilir. Doğrudan proje ile alakalı işlemleri gerçekleştiremez. Doğrudan Sprintlerle alakalı işlemleri gerçekleştiremez. Fakat kendi projesindeki Taskler,Taskloglar ve User'larla ilgili işlemleri gerçekleştirebilir.

- Team Member:

Bir projede temel amacı yönetim kısmı olmayan daha çok Task'leri üzerine alıp çözmesi beklenen kişidir. assignee olarak tanımlandığı Taskler için State değiştirme işlemi yapabilir. Bunun haricinde TaskLog oluşturabilir ve kendi oluşturduğu TaskLog ile işlemleri yapabilir.

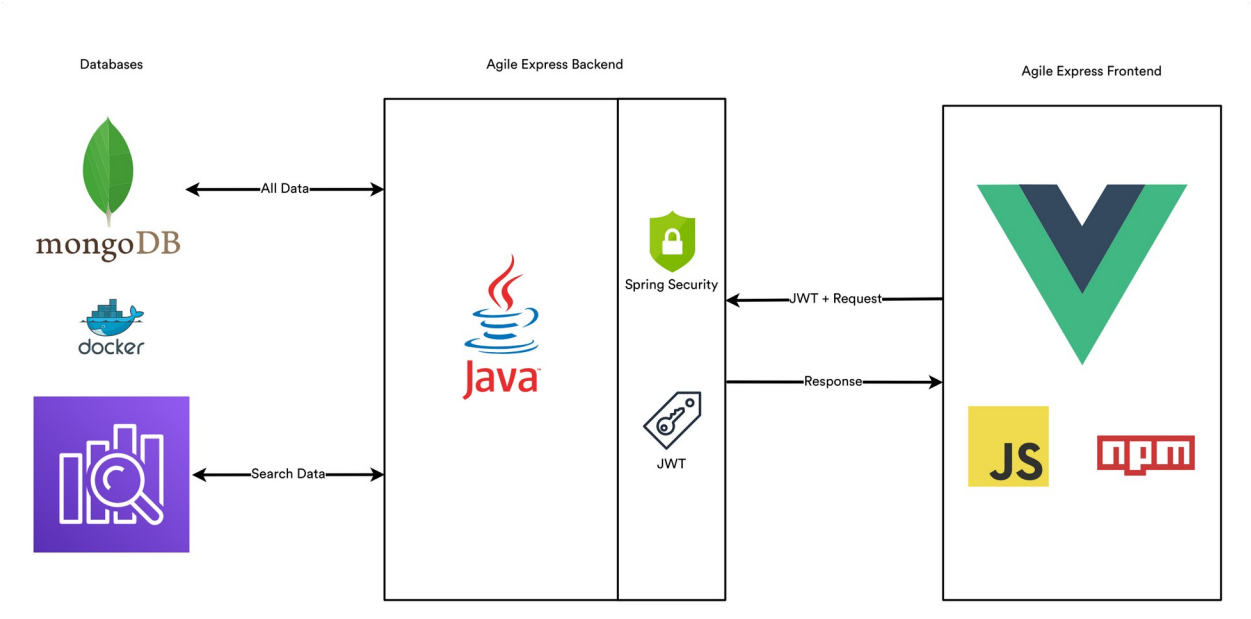
3. Tasarım Detayları

3.1. Temel Tasarım Adımları

Ana mimari 4 farklı yapıdan oluşmaktadır. frontend kod tasarımı, backend kod tasarımı, database tasarımı, search engine tasarımı. Designi daha anlaşılır kılmak için ilerleyen aşamalarda resimler üzerinden gösterimler yapılacaktır.

3.2. Uygulma Çözüm Mimarisi

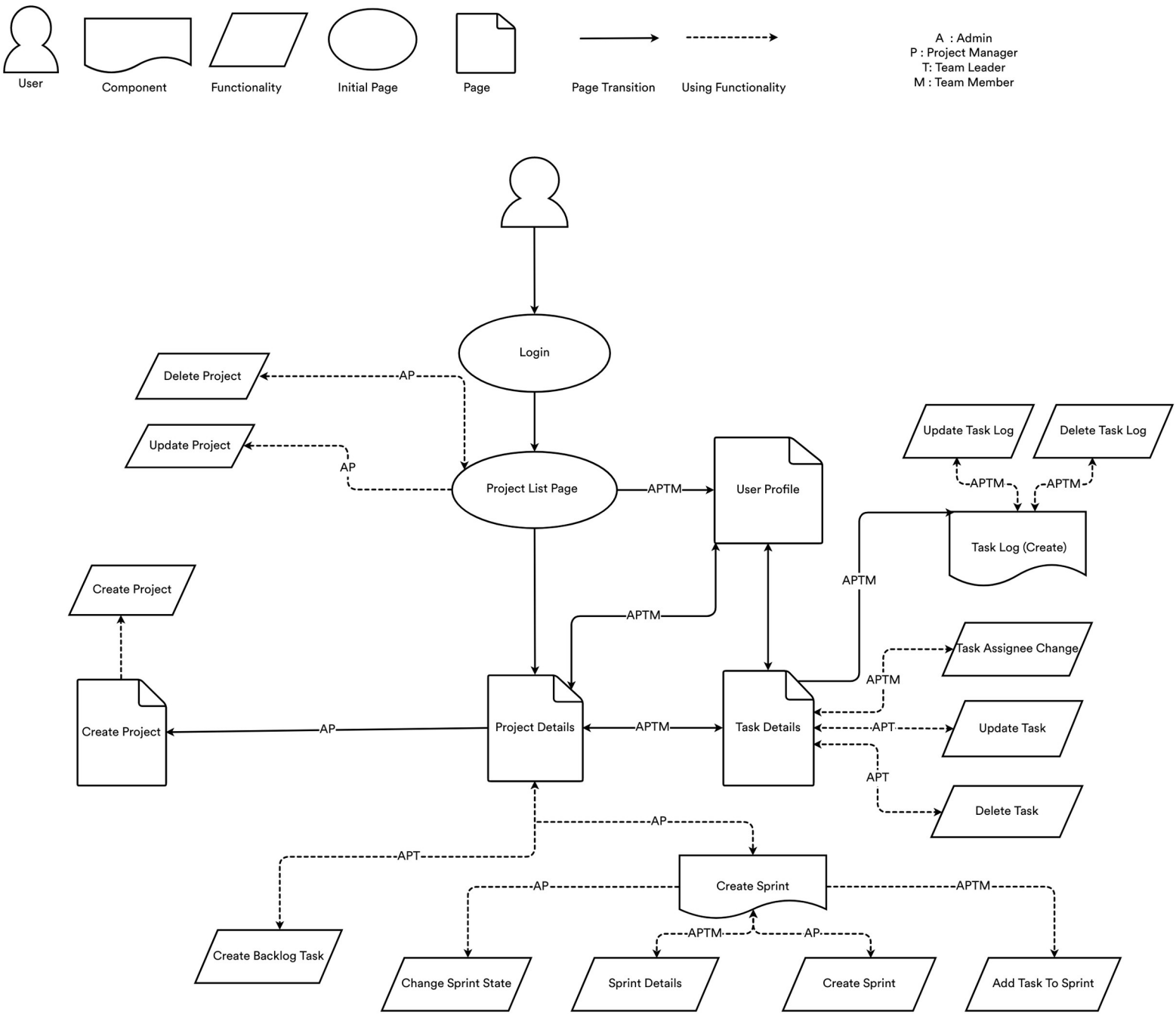
Backend Tarafında LDAP tabanlı bir Authentication yönetimi yapılmıştır. Bunun üstüne frontend auth bilgisi JWT ile sağlanmıştır. Frontend Vue.js ve Node Package Manager üzerinden ilerletilmiştir. Database tarafında Docker-Compose ile başlatılan MongoDB ve ElasticSearch yapıları mevcuttur. Burada Elastic ve Mongo Kod tarafında sync çalıştırılmıştır. Veri tutarlılığı sağlanmaya çalışılmıştır. Fakat yine de Elastic üzerinden kullanıcıya bir geri dönüş yapılmaz. Elastic tarafından gelen datanın MongoDB karşılığı bulunur ve kullanıcıya her zaman MongoDB datası ulaşmış olur.



3.3. Uygulama Mantıksal Diyagramı

Bu kısımda uygulamanın nasıl işlediği, içeriğinde hangi durumların bulunduğu ve hangi kullanıcıların bu durumları gerçekleştirebileceği tasarım üzerinde belirtilmiştir. Üst kısımda hangi sembolün ne işe yaradığı yazmaktadır. Burada tasarımdaki kullanıcı işartei girişi temsil etmektedir.

Tasarımın haricinde Task,Project ve User sayfalarına Search Bar üzerinden erişimde yapılabilir fakat bu global çalıştığı için diğerleriyle ilişkilendirilmemiştir.

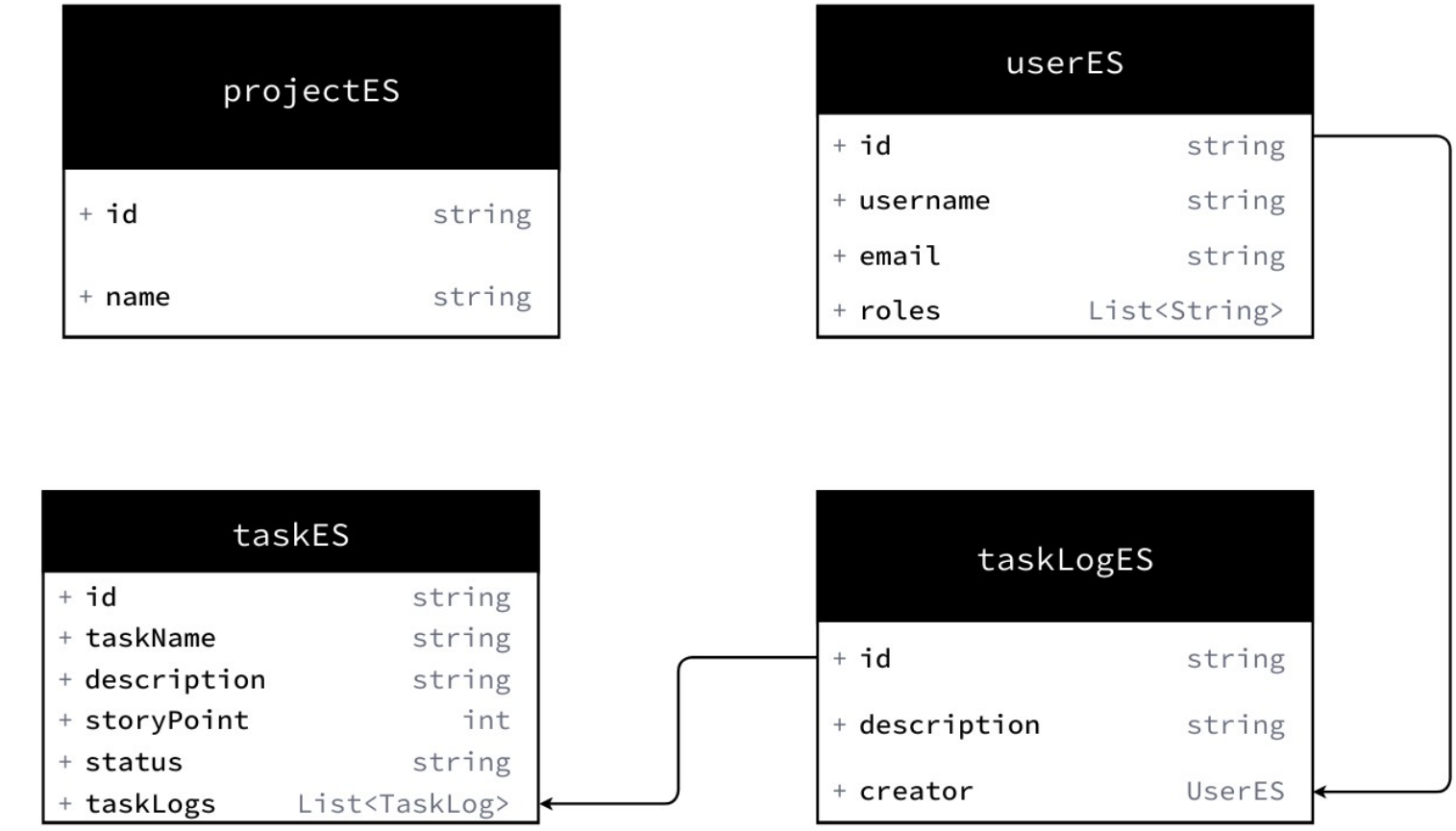


3.4. Fiziksel Deployment Diyagramı (UML Diagram)

Uygulamanın nasıl bir data yapısıyla çalıştığını belirtir. Bu kısımda 2 farklı başlık vardır. Elastic Search Engine tarafında çalışan data yapısı ve uygulamanın database işlevini gerçekleştiren MongoDB'nin data yapısı mevcuttur.

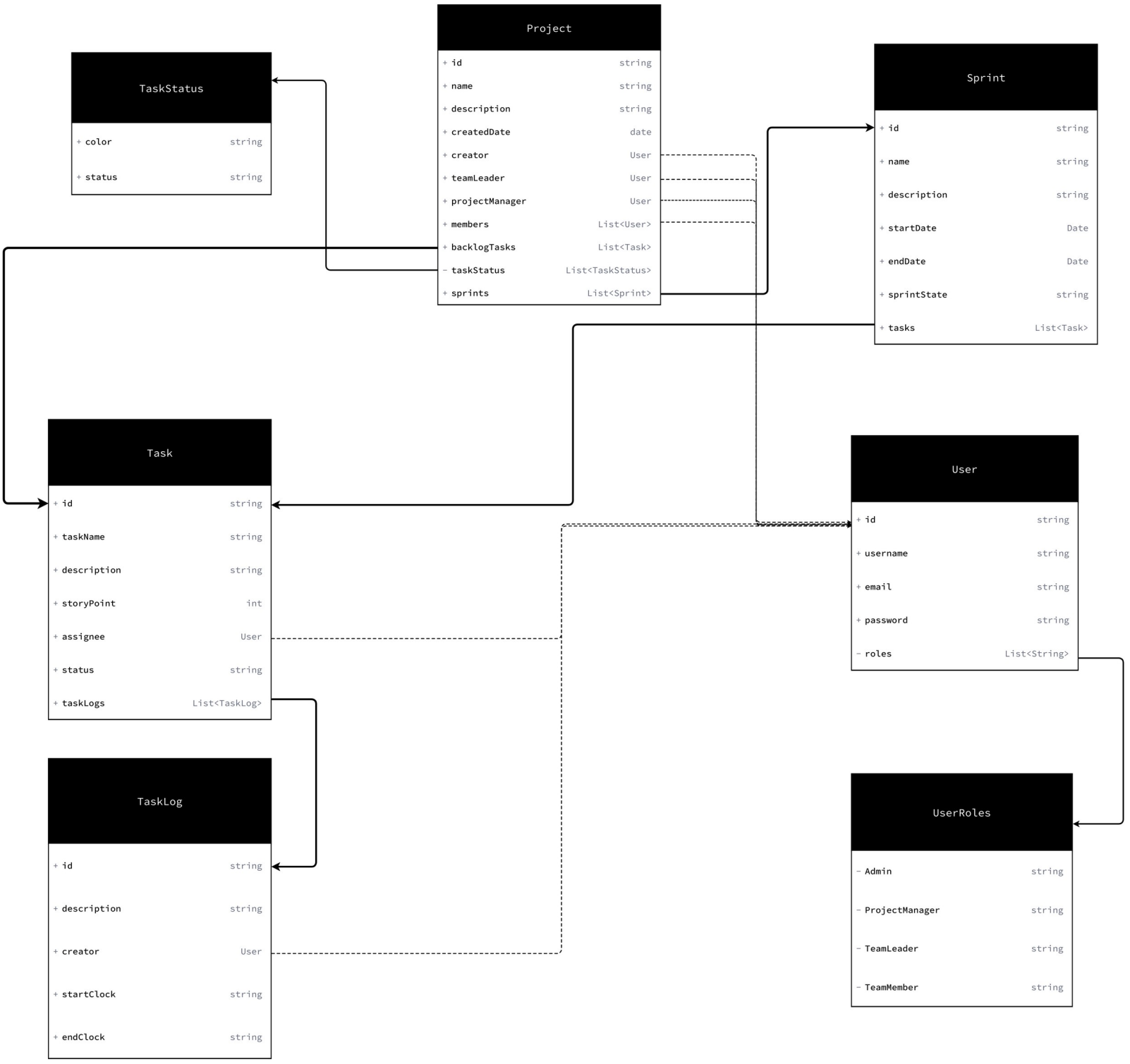
3.4.1 ElasticSearch UML Diagram

ElasticSearch index tabanlı çalışan bir search engine'dir. Burada tuttuğumuz datalar, uygulamanada aslında sadece search tarafında kullanılıyor. Hatta bu datalar bulunduktan sonra mongoDB den tekrar sorgulanıp mongoDB datası kullanıcıya döndürülüyor. Bu yüzden ElasticSearch tarafında relation işlemleriyle uğraşmadan yalnızca aranacak field'lar üzerine bir database kurulmuştur.



3.4.2. MongoDB UML diagram

Burada Documan tabanlı bir veri tabanı söz konusudur. MongoDB tarafında veriler JPA tarafında @DocumentReference yapısı kullanılarak birbirine bağlanmıştır. Burada Relational bir database ile çalışmadığımız için tutarlılık konsunda sıkıntı yaşamamak adına çok fazla many to many ilişkisi kurulmamaya çalışılmıştır. Minimum Relation ile kurulan yapıda kod tarafında sorgular ile gerekli işlemler halledilmiştir.



3.5. Performans

3.5.1. Frontend Performansı

Frontend performansı için Vue.js tarafında lazy-loading routing yapılmıştır. Bu sayede bir component yalnızca ihtiyacımız olduğu durumda çağırılır ve oluşturulur. Bunun dışında JS Bundle şeklinde kullanıcının bilgisayarına yüklenen js dosyaları gerektiği durumlarda çalıştırılır ve kullanıcının bilgisayarında oluşturulan vue template'larının içine daha sonradan REST API istekleri ile data getirilir. Bu sayede veri parçalar halinde gelir ve sayfalar parçalara ayrılmış olur. Server Side Rendering yapılmamış ve performans kazancı sağlanmıştır.

3.5.2. Backend Performansı

Backend tarafında olabildiğince döngülerden kaçınılmış ve karmaşıklık düşürülmeye çalışmıştır. Bunun haricinde ElasticSearch sorguları index tabanlı çalıştığından her ne kadar memory üzerinde yük olsa da işlem hızını oldukça arttırmaktadır.

3.6. Security

Security için Spring Security'nin Controller güvenliği sağlayan @PreAuthorize fonksiyonu ile yetkilere göre api'ler ayrıştırılmıştır. Bunun haricinde Json Web Token ile gelen her isteğin Authorization ve Authenticatin kontrolleri yapılmıştır. Projede CORS yapısı herkese test amaçlı herkese açık şekilde bırakılmıştır fakat config bean'i hazır olduğundan her an değiştirilebilir durumdadır.

3.7. Reliability

Uygulamada herhangi bir database backup sistemi çalışmamaktadır. ElasticSearch tarafında ve MongoDB üzerinde saklanan dataların sycn çalışması kod tarafında kontroller ile sağlanmıştır ve CRUD işlemleri birlikte iletilmektedir.

3.8. Maintainability

Uygulamanın dosya yapısı ve kodun küçük parçalara ayrılarak yazılması, ilerleyen aşamalarda ekleme veya düzenleme gerektiğinde aradığını bulması ve geliştirmesi kolay bir hal kazandırmıştır. Bunun haricinde uygulma 5 farklı domain üzerinden ilerletilebilir. Bunlar : Project,Task,User,TaskLog,Sprint şeklinde ayrılırsa işlem yapmak daha kolay olacaktır.

3.9. Portability

Uygulamada kullanılan Docker containerization yapısı veri tabanlarımızın kendine ait bir işetim sistemi proccessi gibi çalışması ve kendi özgün ortamında çalışmasını sağlar. Bu sayede docker-hub'a yükleyeceğimiz docker file ile farklı bir geliştirme ortamına veya server'a geçtiğimizde ortamdan bağımsız olarak Docker container'ı çalıştırıp stabil database ortamımızı sağlamış olacağız.

3.10. Reusability

Frontend tarafında kullanılan component yapıları (özellikle form componentleri) dinamik bir şekilde çalışacak şekilde tasarlanmıştır. Bu componentler uygulamanın herhangi bir noktasında gerekli datalar gönderilip çalıştırılabilir. Zaten Vue.js kullanmamızdaki en temel özelliklerden bii de budur.