Mentorluk Uygulaması Tasarım Dokümantasyonu

1. Giriş

1.1 Amaç

Bu doküman, geliştirilen sistemin mimari tasarımını kapsamlı bir şekilde açıklamayı amaçlamaktadır.

1.2 Kapsam

Bu doküman, geliştirilen mentorluk uygulamasına mimari bir genel bakış sağlamaktadır. Mentorluk uygulaması sistem tarafından kısıtlanmış konular hakkında danışmanlık vermek isteyen ve danışmanlık almak isteyen kullanıcıları güvenilir bir ortamda buluşturmayı amaçlamaktadır.

2. UML Diyagramları

2.1 Use Case Diyagramı

Bu bölümde kullanıcıların sistemi kullanırken, sistemin sunduğu işlevlerin hangi akışları takip edebileceği açıklanmaktadır. Sistemde iki tip kullanıcı türü yer almaktadır. Bunlar kullanıcılar ve yöneticiler olarak ayrılmakla birlikte kullanıcılar da kendi içerisinde mentor ve mentee olarak iki kullanıcı tipine ayrılmaktadır. Bir kullanıcının mentor olabilmesi için mentorluk başvurusunun yönetici tarafından onaylanması gerekmektedir. Bir mentor, aynı zamanda bir menteedir. Kullanıcılara sunulan işlevler aşağıda listelenmektedir.

- Kullanıcı girişi
  Kullanıcılar kullanıcı adı ve şifreleri veya Google hesapları ile giriş yapabilirler.
- Mentor arama
  Kullanıcılar ana konu veya alt konuya göre mentor filtreleme işlemi gerçekleştirebilirler. Kullanıcılar anahtar kelimeler ile mentor arama işlemi gerçekleştirebilirler.
- Mentorluk talep etme
  Bir kullanıcı mentor olmak istediği ana konu, alt konu seçimlerini yaparak ve seçilen konular ile ilgili deneyimlerini içeren açıklamasını girerek mentorluk başvurusunda bulunabilirler.

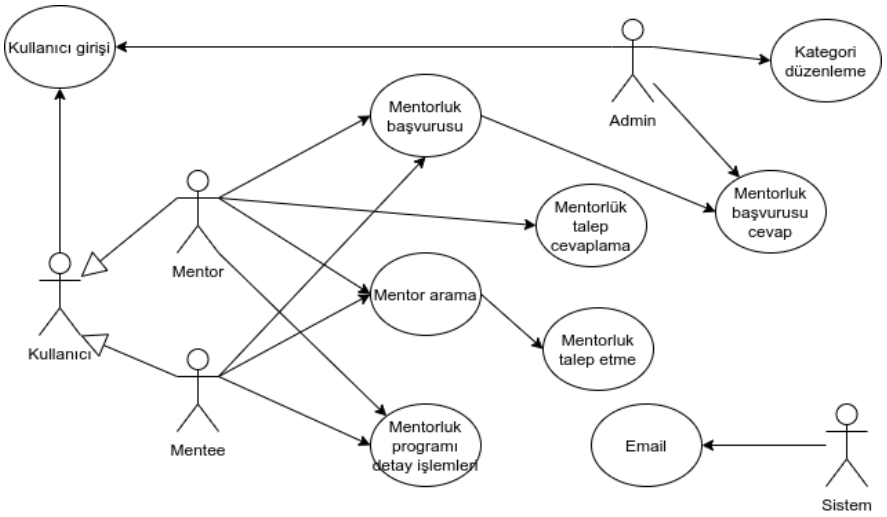
- Mentorluk programının detaylarını oluşturma ve güncelleme
  Mentorluk talep başvurusu onaylanan kullanıcı veya programa dahil olan mentor, programa faz ekleme işlemi, faz sonlandırma işlemi ve sonlandırılmış fazı değerlendirme işlemlerini gerçekleştirebilir.
- Kayıtlı olduğu mentorluk programlarını listeleme
  Kullanıcılar mentor veya mentee olarak dahil oldukları mentorluk programlarını ana sayfalarında görüntüleyebilirler.
- Yapmış olduğu mentorluk başvurularını listeleme
  Kullanıcılar yapmış oldukları mentorluk başvurularının durumlarını görüntüleyebilirler.

Yalnızca mentor kullanıcı tipine sahip olan kullanıcılara sunulan işlevler ise şu şekildedir.

- Mentorluk talep yanıtlama
  Kullanıcılar kendilerinden talep edilen mentorluk programını kabul edebilir veya reddedebilirler.

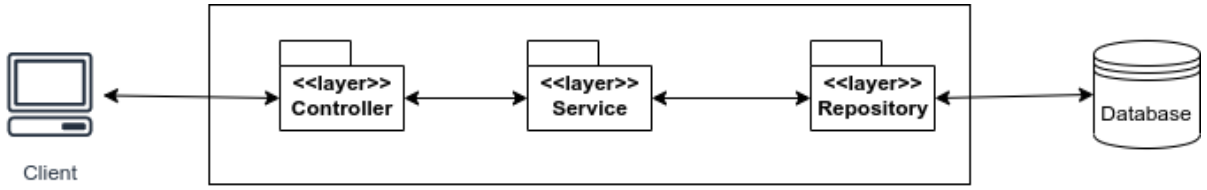
Yönetici kullanıcı tipine sahip kullanıcılara sunulan işlevler aşağıda listelenmektedir.

- Kategori ve konu oluşturma, düzenleme ve silme işlemleri
  Yöneticiler ana konu ve alt konular üzerinde oluşturma, güncelleme ve silme işlemlerini gerçekleştirebilirler.
- Mentorluk başvuru taleplerinin listelenmesi ve cevaplanması
  Yöneticiler, kullanıcılar tarafından oluşturulan mentorluk başvurularını kabul edebilir veya reddedebilirler.

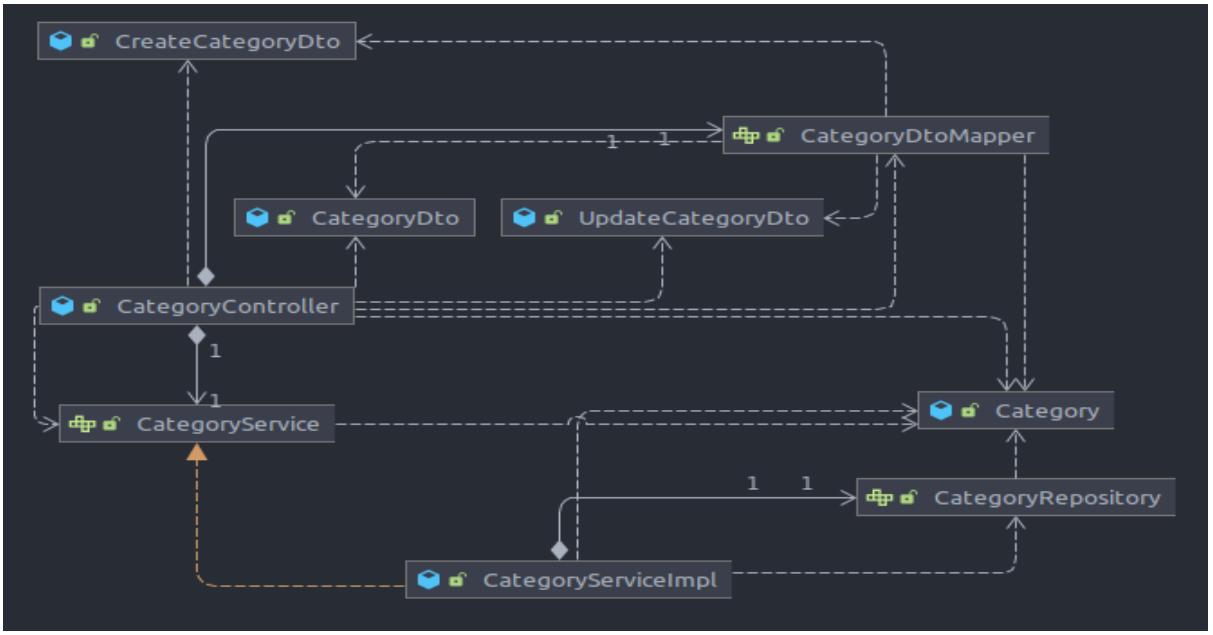


2.2 Class Diyagramları

Her bir servis tasarlanırken Multi Layered Architecture tasarım kalıbı kullanılmış ve servis dışından alınan veya servis dışına gönderilen veriler için DTO (Data Transfer Object) yapısı kullanılmıştır.



- Kategori servisine ait class diyagramı

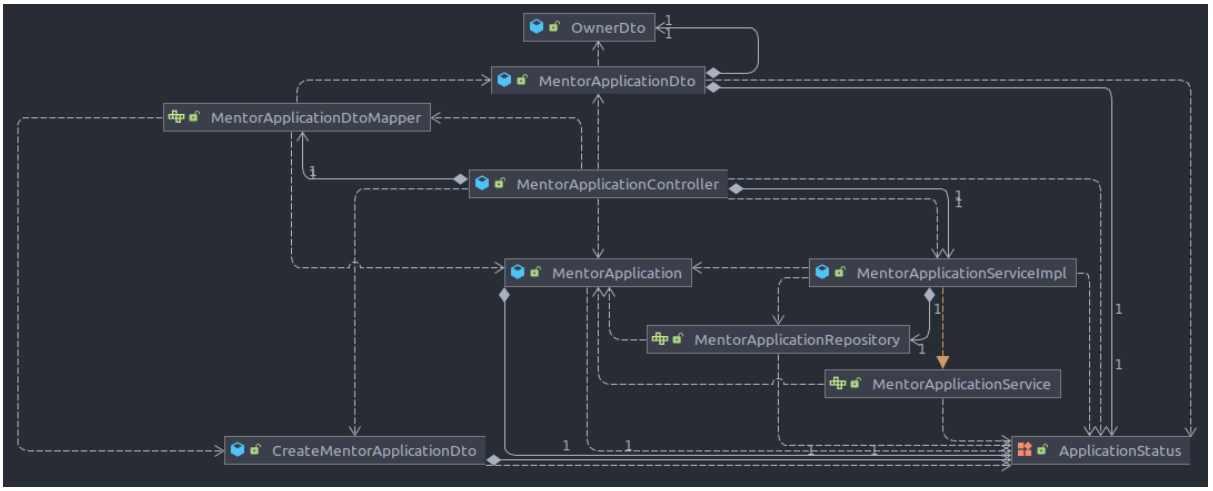


CategoryController sınıfı sunucuya dışarıdan gelen istekleri karşılamaktadır. Gelen isteklerin türüne göre istekle gelen objeyi Dto modellerinden uygun olana dönüştürmektedir. Yine CategoryController içerisinde, alınan Dto objesi CategoryDtoMapper interface'i yardımıyla Category entity modeline dönüştürülmekte ve servis katmanında bulunan CategoryServiceImpl sınıfına aktarılmaktadır. Servis katmanında gerekli işlemler yapıldıktan sonra controller katmanına yapılan işlem sonucunda oluşan entity modeli dönülür. Controller katmanı tekrar bir dönüşüm yaparak entity objesini CategoryDto objesine çevirir ve isteği yapmış olan istemciye cevabını gönderir.

Servis katmanında ise controller katmanından alınan entity Repository katmanına iletilir. Repository katmanı aldığı veri ile istenilen işlemi veritabanı üzerinde gerçekleştirir ve sonucu servis katmanına iletir.

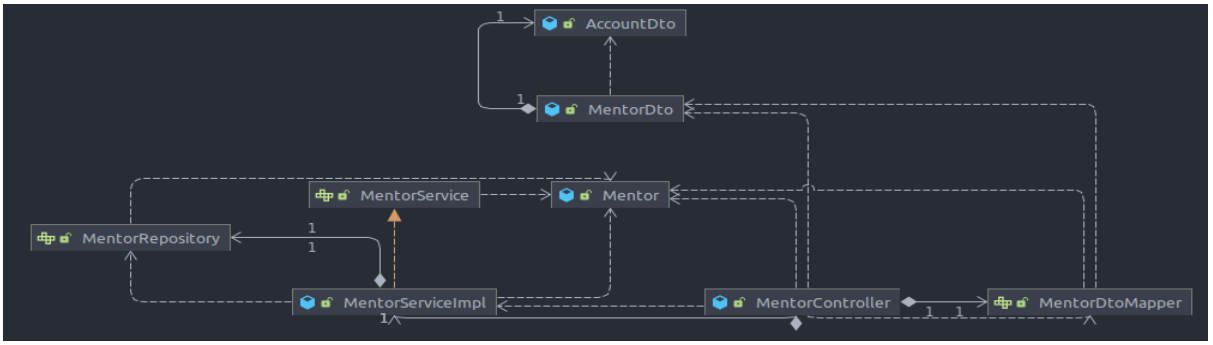
Yukarıda anlatılan çok katmanlı mimari tasarımın çalışma biçimi diğer servislerde de aynı olduğundan ilerleyen başlıklarda dokümanın kolay okunabilirliği açısından tekrar ele alınmayacaktır.

- Mentorluk başvurusu servisi class diyagramı



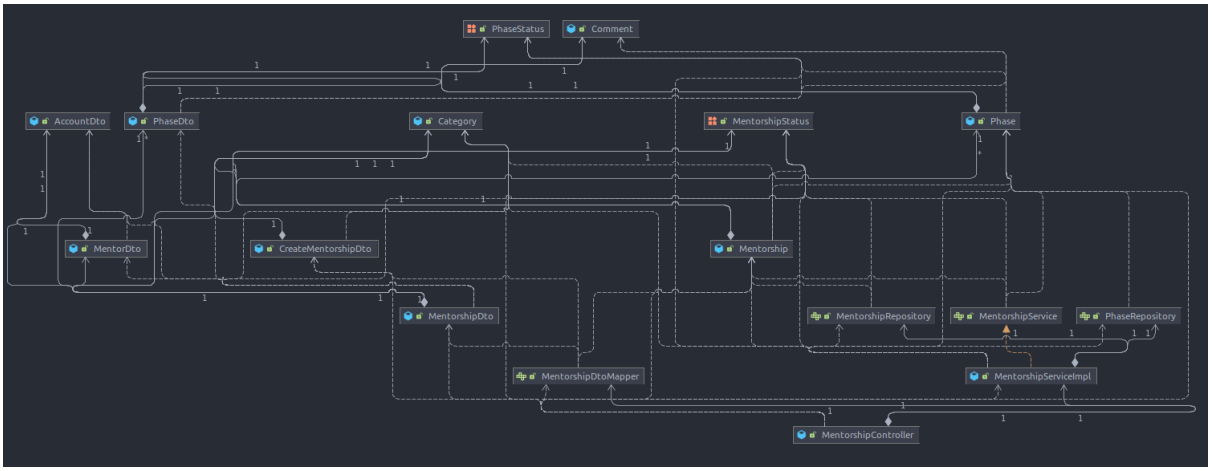
Mentorluk servisi, mentorluk başvurularının oluşturulduğu ve güncellendiği servistir. Ayrıca başvuruları kabul edilen kullanıcılar için bir mentor kaydı ve mentorluk postu oluşturur.

- Mentor servisi class diyagramı



Mentor servisi, yalnızca sistemde kayıtlı olan mentorların listelenmesi için kullanılmaktadır.

- Mentorship servisi class diyagramı

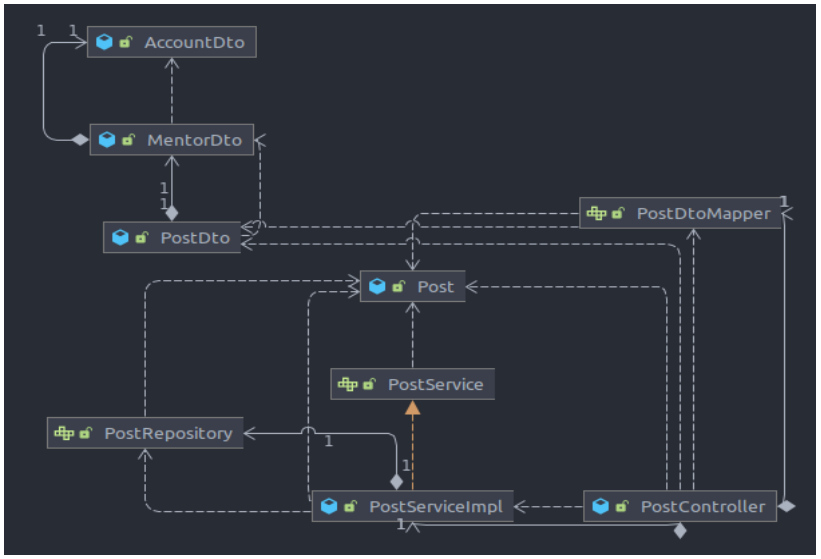


Mentorship servisi diğer servis hem diğer servislere bağımlılığı (mentor ve email servisleri) hem de kendi içerisinde 2 adet entity objesi ve bu entitylerin Repository nesnelerine bağımlılık içerdiği için diğer servislere göre daha kompleks bir görünüm sergilemektedir.

Mentorship servisi, bir mentorun aynı anda yalnızca iki adet mentorluk programına dahil olabilmesi kısıtından dolayı mentorun uygunluk durumunu güncelleyebilmek için MentorRepository objesini içermektedir.

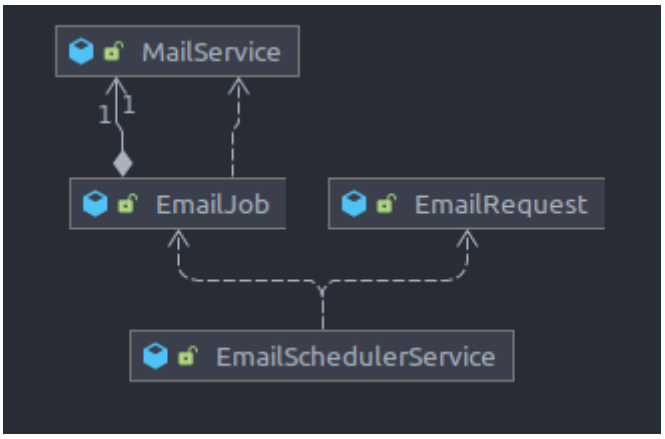
Mentorluk programına ait fazların oluşturulması ve güncellenmesi işlemleri yine bu servis tarafından gerçekleştirilmektedir. Bir mentorluk programına ait fazlar oluşturulurken fazın bitiş tarihine göre zamanlanmış e-mail istekleri oluşturulmaktadır. Bu nedenle bu servisin e-mail servisine de bir bağımlılığı mevcuttur.

- Post servisi class diyagramı



Post servisi, mentorların danışmanlık verdikleri konuları ve açıklamalarını içeren ve diğer kullanıcılar tarafından arama, filtreleme gibi işlemleri gerçekleştirebildiği post işlemlerini sağlamaktadır.

- E-mail servisi class diyagramı



Bu servis bir mentorluk programına dahil olan kullanıcılara, oluşturdukları fazların sonlanma tarihine bir saat kalınca bildirim e-maili göndermek için zamanlanmış işlemleri gerçekleştirir.

- Security servisi

Security servisi, geliştirilen bu sistem içerisinde en kompleks yapıyı barındıran servistir. Yaptığı işlem en basit seviyede, embedded LDAP server ve OAuth2 mekanizmalarını kullanarak sunucuya gelen istekleri doğrulamak ve rol bazlı olarak gelen isteğin yapılmak istenen işleme yetkisinin olup olmadığını kontrol etmek şeklinde tanımlanabilir.

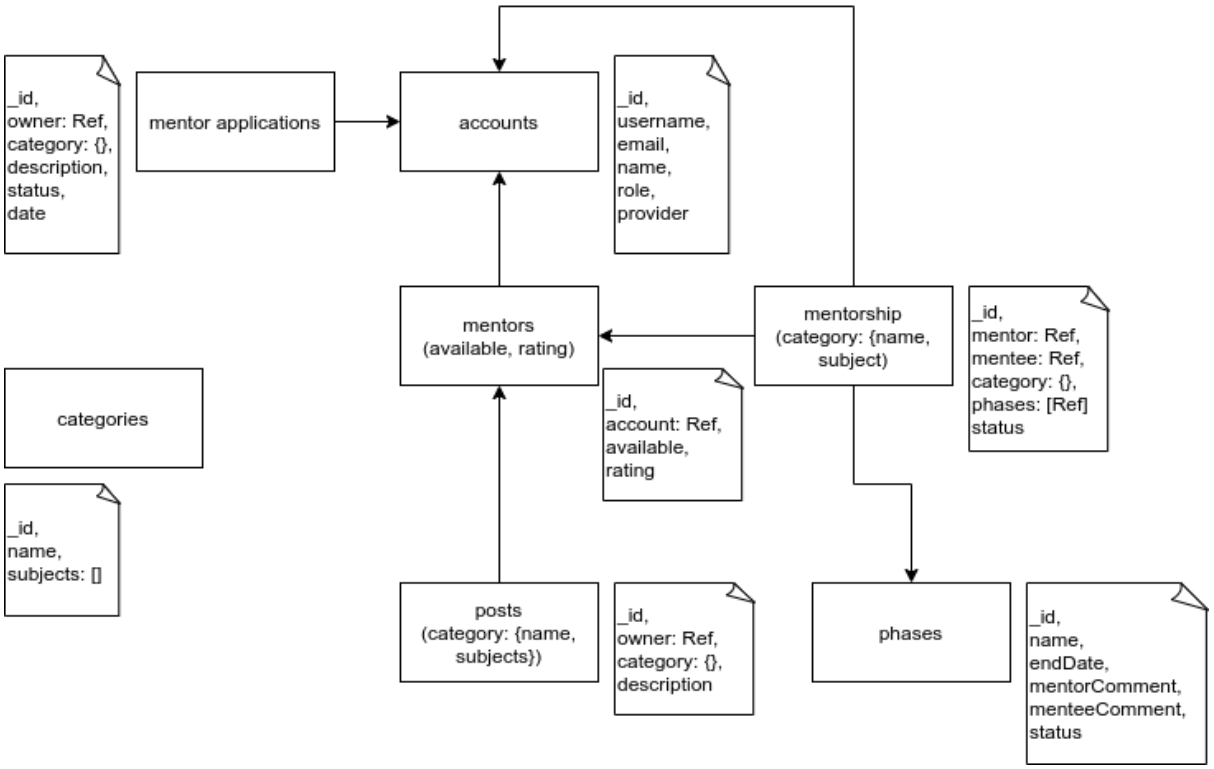
LDAP kullanılarak doğrulanan kullanıcılar için güvenlik mekanizması olarak HttpBasic security kullanılmıştır. Kullanıcı bir kez doğrulandıktan sonra istemciye, kullanıcı için oluşturulan session id spring security framework tarafından gönderilir. Kullanıcı daha sonraki istekleri için bu session id ile doğrulanır.

OAuth2 kullanılarak yapılan doğrulama için ise sunucu tarafından oluşturulan JWT token doğrulanan istemciye gönderilir. LDAP için kullanılan güvenlik mekanizması bir session'a ihtiyaç duyduğu için OAuth2 ile yapılan doğrulama sırasında da kullanıcıya otomatik olarak bir session id iletilmektedir.

LDAP ve OAuth2 yalnızca kullanıcı ve yetki doğrulamak için kullanılmaktadır. Kullanıcı bilgileri ayrıca bir veritabanında tutulmaktadır.

3. Veri Tabanı

Sistemde ana veritabanı olarak bir NoSQL veritabanı olan MongoDB kullanılmaktadır. Koleksiyonlar arası ilişkiler mümkün olduğunca embedded document yöntemi ile sağlanmaya tasarlanmıştır.. Ancak one-to-few ilişkiden one-to-many ilişkiye dönüşebilecek ilişkiler veya bazı özel koleksiyonlar document reference yöntemi ile tasarlanmıştır. Sistemde bulunan verilerin ilişkileri aşağıdaki şemada belirtilmiştir.



Veritabanı tasarımında öncelikle kullanıcıların hesap bilgilerinin tutulacağı koleksiyon tasarlanmıştır. Bu koleksiyon kullanıcı adı, isim, e-mail, kullanıcı tipi ve kullanıcı doğrulaması için kullanılan yöntem bilgilerini içermektedir.

Bir kullanıcının mentorluk başvurusu kabul edilir ise kullanıcı ayrıca mentor koleksiyonuna kaydedilir. Mentor koleksiyonu, hesaplar koleksiyonundaki kullanıcı bilgilerini referans olarak tutmaktadır. Ayrıca mentorun uygunluk durumunu kontrol etmek için ayrıca bir “available” alanı içermektedir.

Benzer şekilde bir kullanıcının mentorluk başvurusu kabul edilir ise mentor bilgilerini, mentorluk verilecek konuyu, alt konuları ve mentorun açıklamalarını içeren bir kayıt post koleksiyonuna kaydedilir. Burada mentor bilgisi yine referans değeri olarak tutulmaktadır.

Bir kullanıcı, sistemde postu yayınlanmış bir mentorden mentorluk talep ettiğinde bu başvuru bilgisi mentorluk programı (mentorship) koleksiyonuna durumu bekliyor (PENDING) olarak

kaydedilir. Eğer program sahibi olan mentor başvuruyu onaylar veya reddederse ilgili kayıdın durum bilgisi mentorun cevabına göre güncellenir.

Mentor tarafından onaylanmış bir mentoluk programına mentor veya mentee tarafından fazlar eklenebilir. Bu fazlar mentorluk programı (mentorship) koleksiyonunda referans dizisi olarak tutulur. Fazlar faz ismini, fazın bitiş tarihini, mentor ve menteenin değerlendirmelerini içermektedir.

Sistemdeki konular ve alt konular ise kategoriler koleksiyonunda tutulmaktadır.

Zamanlanmış işlemlerin kaydedilebilmesi için MySQL ilişkisel veritabanı kullanılmıştır.

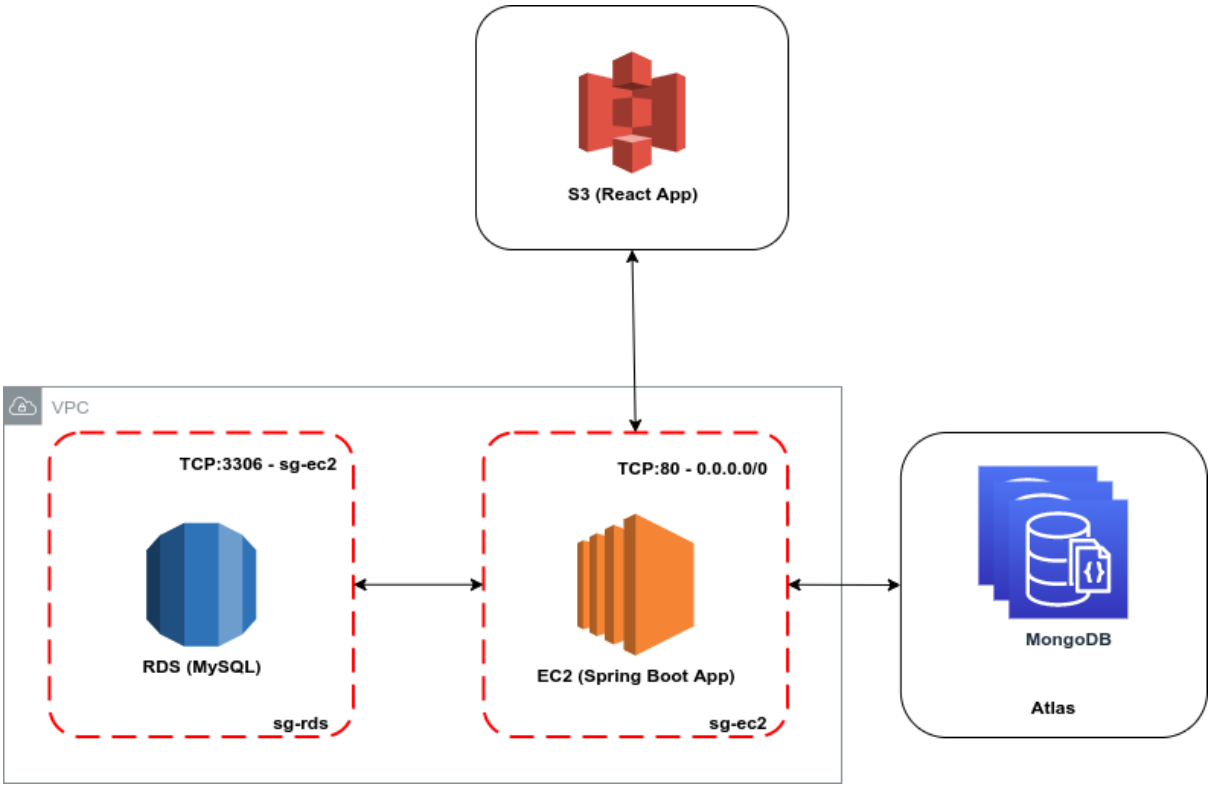
4. Deployment Diyagramı

Sistemin ana veritabanı olarak kullanılan MongoDB veritabanı MongoDB'nin cloud çözümü olan atlas üzerinde üç farklı instance ile çalışmaktadır.

Sistemin web sunucusu, AWS üzerinde bir EC2 instance içerisinde çalışmaktadır.

MySQL veritabanı ise yine AWS üzerinde RDS servisinde internete kapalı olarak yalnızca sistemin web sunucusu tarafından erişilebilir şekilde çalışmaktadır.

Sistemin istemci tarafı ise AWS S3 bucket üzerinde çalışmaktadır.



5. Çözüm Mimarisi

Sistemin genel tasarımı için server-client mimarisi uygun görülmüş ve bu doğrultuda web sunucusu ve client için ayrı uygulamalar geliştirilmiştir.

Web sunucusu tarafında Java ekosisteminde en çok tercih edilen frameworklerden biri olan Spring Framework tercih edilmiştir. Bu tercihin sebeplerinden bazıları Spring Framework'un bir web sunucusunun ihtiyaç duyduğu bir çok konuda çözümlerinin olmasıdır. Web sunucusunun client ile iletişimi için Restfull mimarisinden faydalanılmıştır.

Bunlardan ilki otomatik konfigürasyon sağlayarak bir web sunucunun daha hızlı bir şekilde ayağa kaldırılmasına yardımcı olan Spring Boot'dur.

Authentication ve authorization işlemleri için bir çözüm sunan Spring Security ve veritabanı erişim katmanını Spring Data ile soyutlayarak kullanıma hazır çözümler sunması bir diğer tercih sebebi olmuştur.

Veritabanı seçiminde önce tasarlanan veritabanının yapısal verilere sahip olacağı düşünülmüş ve bir ilişkisel veritabanı olan PostgreSQL tercih edilmiştir. Daha sonra bazı ilişkilerin gereksiz olabileceği göz önünde bulundurularak doküman bazlı bir veritabanı olan MongoDB seçimine dönülmüştür. Ayrıca sistem gereksinimlerinden biri olan mentor arama özelliğinin MongoDB'nin sunduğu text index özelliği ile karşılanabiliyor olması da yapılan seçim için bir etken olmuştur.

Arayüz geliştirme için ise en popüler JavaScript frameworku olan React.js tercih edilmiştir. Giriş yapmış olan kullanıcının bilgilerini global olarak saklayabilmek için redux ve redux-saga kullanılmıştır.