SAMSUN ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ
2023 – 2024 Akademik Yılı Güz Dönemi
MYAZ201 – Yazılım Gereksinimi ve Modelleme Dersi Final Sınavı

18/01/2023

Değerli öğrencilerim,

MYAZ201 Yazılım Gereksinimi ve Modelleme dersinin final sınavına hoş geldiniz. Sınav boyunca uyulması gereken kurallar aşağıda listelenmiştir. Sınavda uyulması gereken kuralları lütfen dikkatle okuyunuz.

- Sınav süresi 90 dakikadır.
- Sınav için kullanılması gereken materyaller (dokümanlar ve kaynak kodlar vb.) öğrenme yönetim sistemi üzerinden indirilmelidir.
- Sınav süresince internet ve yapay zekâ destekli araçların kullanımı yasaktır.
- Sınav süresince bilgisayar kullanabilirsiniz. Kaynak kodlar, editör kullanımı ve ders notlarına erişim sağlayabilirsiniz.
- Sınav başladıktan sonra, sınav bitimine kadar sınav salonlarına giriş/çıkış yapılmayacaktır.

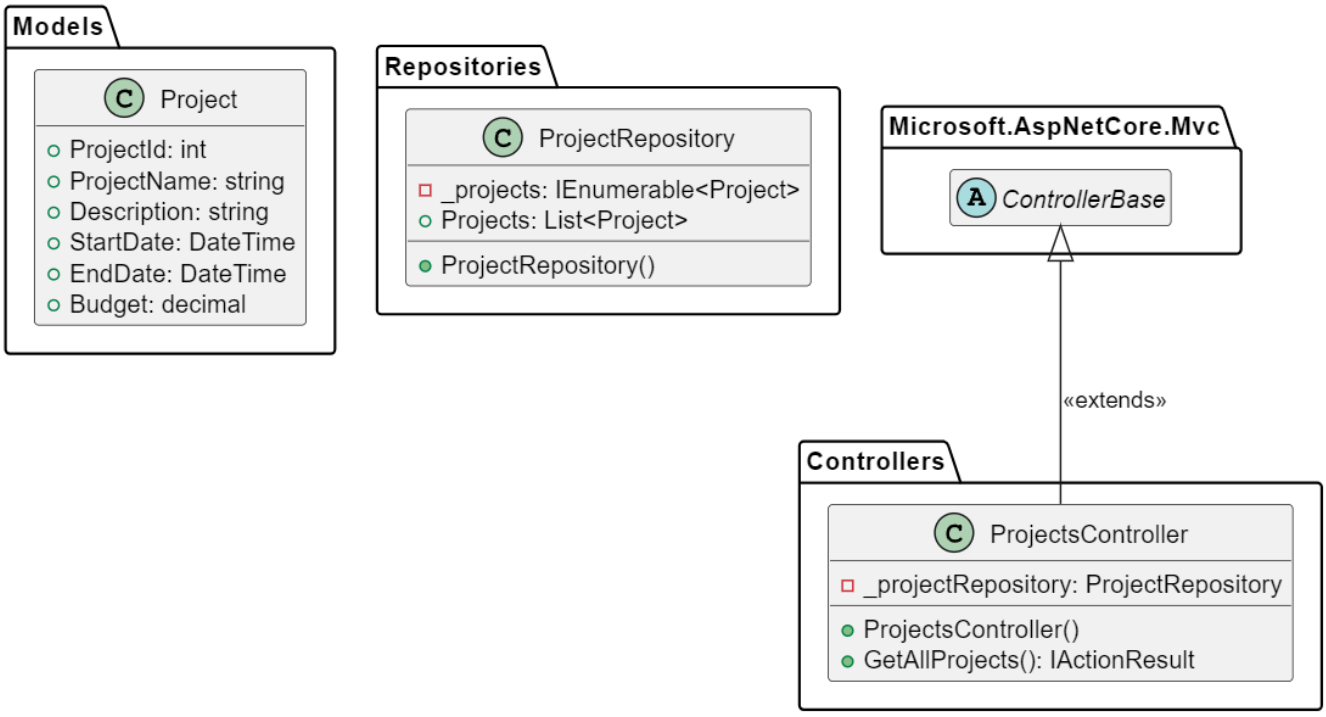
Cevap Kağıdının Hazırlanması

- Cevap kâğıdınızın sağ üst köşesine:
  - Adınızı soyadınızı,
  - Öğrenci numaranızı
  - **Şube numaranızı**

  mutlaka yazınız.
- Sınavda sadece sizden istenen ifadeleri cevap kâğıdınıza geçiriniz. Tanımı istenmeyen ifadeleri yazmayınız. **Tek bir cevap kağıdı kullanma noktasında azami özen gösteriniz.**
- Birden fazla cevap kâğıdı kullanmanız durumunda lütfen ilgili cevap kâğıtlarını numaralandırınız ve sınav görevlisinden cevap kâğıtlarını zımbalamasını isteyiniz.
- Cevap kağıdınızda her bir soruyu net bir şekilde birbirinden ayırınız.

SORULAR

1. Proje yönetimi için bir API tasarımı gerçekleştirilmiştir. Bu çerçevede tasarlanan yapıya ait UML diyagramlarına aşağıda yer verilmiştir.



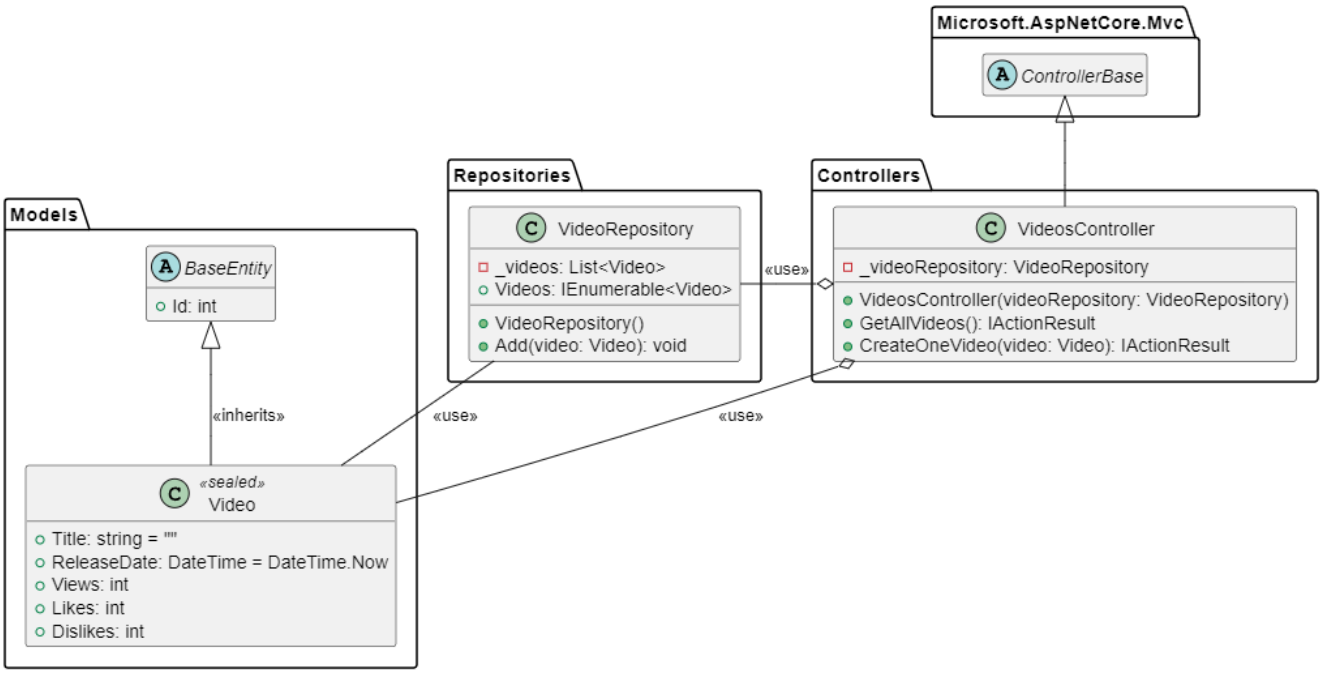
İsterler

- `./api/project/less-than-budget/{budget}` adresine bir `GET` isteği atıldığında `{budget}` parametresinde belirtilen değerden küçük bütçeli projelerin listelenmesi beklenmektedir.
- `{budget}` değeri sıfır ya da negatif olamaz. `{budget}` değerinin sıfır ya da negatif olması durumunda ilgili uygulamanın hata fırlatması ve `400 Bad Request` kodu ile dönüş yapması beklenir.

Cevap Kağıdına Geçilmesi Gerekenler

- `ProjectsController.GetFilteredProject()` metodunun gövdesi (10 p).

2. Video yönetimi için bir API tasarımı gerçekleştirilmiştir. Bu çerçevede tasarlanan yapıya ait UML diyagramlarına aşağıda yer verilmiştir.



İsterler

- `api/videos` adresine `POST` isteği ile gidildiğinde yeni bir kaynak oluşturulmalıdır.
- Oluşturulmak istenen kaynağın `Id` değeri listedeki kayıt sayısına +1 eklenerek atanmalıdır.
- Oluşturulmak istenen kaynak boş ya da geçerli değilse uygulama `400 Bad Request` ile dönüş yapmalıdır.
- Kaynak başarılı bir şekilde eklendiyse uygulama `201 Created` kodu ile dönüş yapmalı ve `Reponse` içerisinde `Header` bölümüne `location` anahtar değerini ekleyerek oluşturulan kaynağın bilgisi cevaba eklenmelidir.
- `VideoRepository` nesnesi tüm kullanıcılar için ortak olarak kullanılabilmelidir. Tüm kullanıcılar için bu nesneden yalnızca tek bir örnek türetilmelidir.

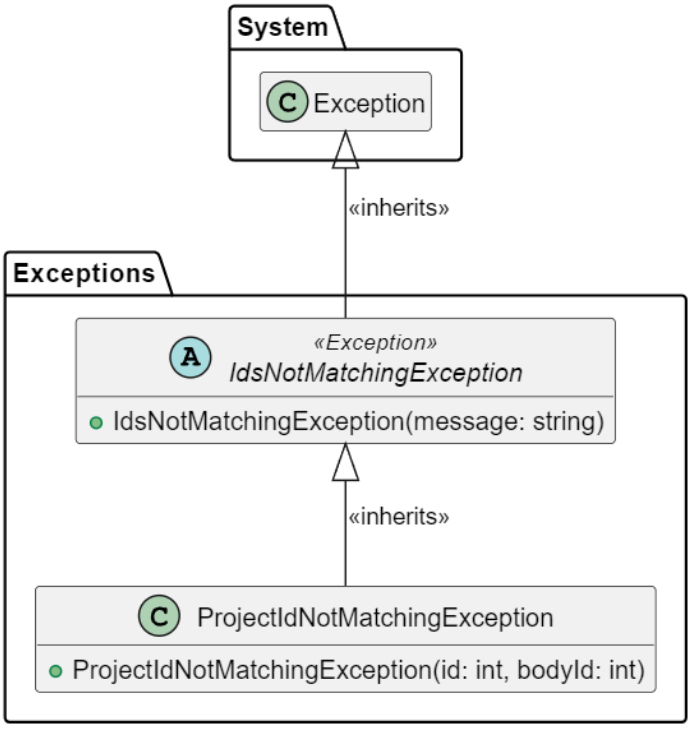
Cevap Kağıdına Geçilmesi Gerekenler

a) `VideoRepository.Add(Video)` metodu (10p).

b) `VideosController.CreateOneVideo(Video)` metodu (10p).

c) `IoC` kaydı (`register`) için yaptığınız tanımları cevap kağıdınıza geçiriniz (10p).

3. Proje yönetimi için bir API tasarımı gerçekleştirilmiştir. API veri tabanlı ile çalışacak hale getirilmiştir. Bu uygulamada hem güncelleme hem de hata yönetimi gerçekleştirmeniz beklenmektedir. Uygulamaya mimarisi önceki sorulardaki yapıya benzer şekilde organize edilmiştir. Hata yönetimi için tanımlanması gereken ifadelere ait UML diyagramına aşağıdaki şekilde yer verilmiştir.



İsterler

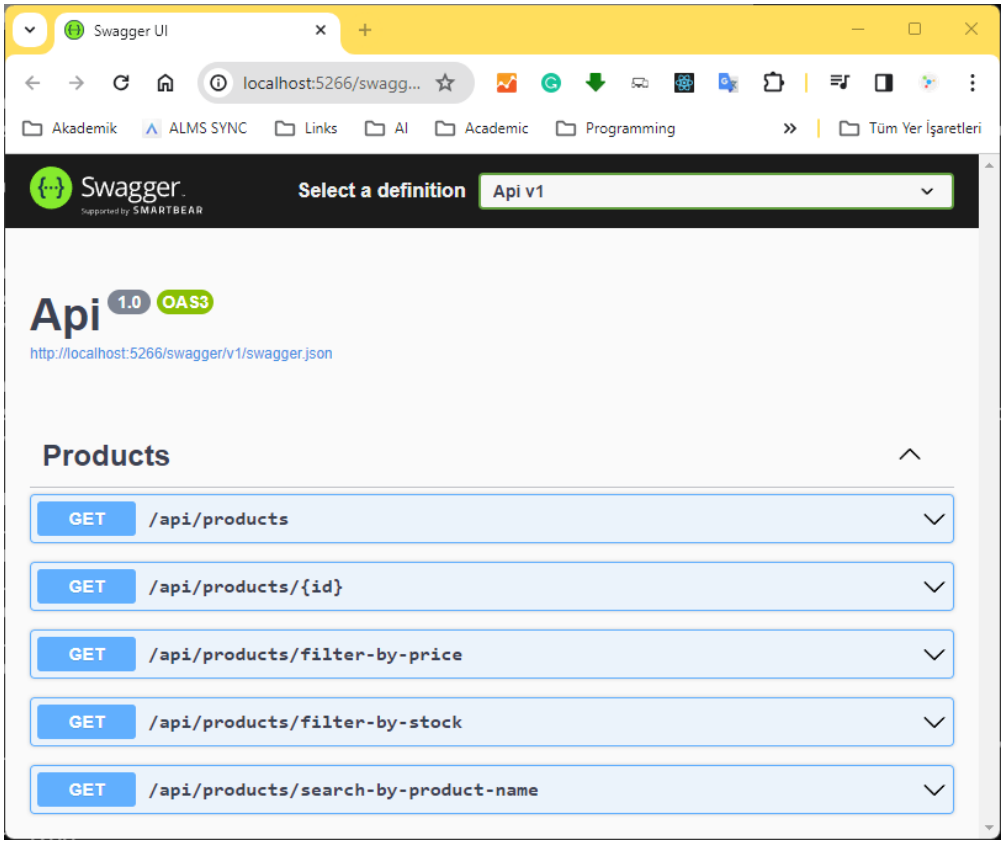
- `api/projects/{id}` endpoint noktasına bir `PUT` isteği geldiğinde `{id}` değeri verilen projenin güncellenmesi istenmektedir.
- Güncellenmek istenen kaynak `Request Body` ile gönderilmelidir.
- Parametre olarak gelen `{id}` değeri ile metot gövdesindeki `{id}` değerinin eşleşmemesi durumunda `ProjectIdsNotMatchingException(int,int)` hatası fırlatan bir özel bir hata türü yazmanız beklenmektedir. Bu hata türü ile ilgili yapılması gereken tanımlara UML diyagramında yer verilmiştir.

Cevap kağıdına geçirilmesi gerekenler

a) `ProjectRepository.UpdateOneProject(int,Project)` metodunun gövdesi (10p).

b) `ProjectIdsNotMatchingException` sınıfının tamamı (10p).

4. Ürün yönetimi için bir API tasarımı gerçekleştirilmiştir. API `Sqlite` veri tabanıyla çalışabilecek şekilde tasarlanmıştır. Uygulama mimarisi önceki sorularda verilen mimariye benzerdir.



İsterler

- Bu API bilgisayarınızda çalışacak hale getiriniz. Bu çerçevede Code First yaklaşımı gereği Migrations işlemlerini öncelikle tamamlayınız.
- `api/products/filter-by-price` endpoint gelen isteği karşılayacak metot tasarımını gerçekleştiriniz.
  - İstek başarılı bir şekilde yapılırsa, `200 OK` durum kodu ile birlikte belirtilen fiyat aralığına sahip ürünlerin listesi döner.
  - Filtreleme sonucunda hiç ürün bulunamazsa, boş bir liste (`[ ]`) döner.
  - Eğer `minPrice` veya `maxPrice` parametreleri eksikse, `400 Bad Request` durum kodu ile birlikte uygun bir hata mesajı döner. Örneğin, "***Geçerli bir minPrice ve maxPrice belirtmelisiniz.***"
  - Eğer `minPrice` parametresi, `maxPrice` parametresinden büyükse, `400 Bad Request` durum kodu ile birlikte uygun bir hata mesajı döner. Örneğin, "***minPrice, maxPrice'den büyük olamaz.***"

- Örnek
  - `/api/products/filter-by-price?minPrice=10&maxPrice=50` gibi bir istekle, fiyat aralığına sahip ürünleri filtreleyebilir ve bu ürünleri alabilirsiniz.
  - `/api/products/filter-by-price?minPrice=50&maxPrice=10` gibi bir istekle, geçersiz bir fiyat aralığı belirtilmiştir. Bu durumda, `400 Bad Request` durum kodu ile birlikte uygun bir hata mesajı alabilirsiniz.

- `/api/products/search-by-product-name?searchTerm=apple` gibi bir istekle, belirtilen kelimeyi içeren ürünleri alabilir ve ürün adına göre birden fazla ürün varsa alfabetik sıralamalısınız.
  - İstek başarılı bir şekilde yapılırsa, `200 OK` durum kodu ile birlikte belirtilen kelimeyi içeren ürünlerin alfabetik sıralı listesi döner.
  - Filtreleme sonucunda hiç ürün bulunamazsa, boş bir liste (`[]`) döner.
  - `/api/products/search-by-product-name` gibi bir istekle, eksik bir `searchTerm` parametresi belirtildiği için `400 Bad Request` durum kodu ile birlikte uygun bir hata mesajı alabilirsiniz.

Cevap kağıdına geçirilmesi gerekenler

- `ProductsController`, alanlar (`fields`), özellikler (`properties`) ve yapıcı metotlar (`constructors`) tanımları (10p).
- `api/products/ filter-by-stock` uç noktasını karşılayan metodun gövdesi (10p).
- `api/products/search-by-product-name` uç noktasını karşılayan metodun gövdesi (10p).

5. Aşağıdaki ifadelerde bu bırakılan yerleri uygun ifadelerle tamamlayınız. Cevap kağıdınıza sadece (a), (b), (c), (d) ve (e) alanlarına girdiğiniz ifadelerin Türkçe ve İngilizcelerini yazınız (10p).

Representational State Transfer (Temsil Durumu Transferi), **REST** olarak ifade edilir ve web tabanlı uygulamalar arasında iletişim kurmak için kullanılan bir mimari stildir. Bu mimari stili, Roy Fielding tarafından 2000 yılında doktora tezi olarak tanıtılmıştır. **REST**, özellikle **HTTP** protokolü üzerinde çalışır ve web servisleri oluşturmak için popüler bir yaklaşımdır. REST mimarisinde her şey ___(a)____ etrafında döner. ___(a)____ birbirinden ayırt etmek üzere **Unified Resource Identifier (URI)** yapıları, bir başka ifadeyle **endpoint**'ler kullanılır.

İstemci, sunucudan bir kaynağın temsilini alır (**Representational State Transfer**). Bu temsil genellikle **XML** veya **JSON** formatında olabilir. İstemci, aldığı temsil üzerinden kaynağın durumuyla etkileşimde bulunur. Her istek, sunucu tarafından bağımsız olarak işlenir ve her istek, gerekli tüm bilgiyi içermelidir. Sunucu, istemcinin durumunu saklamaz. Bu ____(b)_____ kavramı ile ifade edilir.

İstemciler, sunucudan aldıkları temsil üzerinden ilerleyebilirler. Yani, sunucu, istemciye bir kaynağın durumu hakkında bilgi verir ve istemci bu bilgileri kullanarak diğer kaynaklarla etkileşime geçer. ____(c)____ bir RESTful web servisi tasarlarken kullanılan bir prensiptir. Bu prensip, istemcilerin sunucuyla etkileşimde bulunurken, sunucudan alınan temsil (**representation**) üzerinden kaynaklar arasında gezinmelerini sağlamak amacını taşır. Temelde tasarlanan API'nın keşfedilebilir olmasını sağlar.

HTTP isteğinde "**Accept**" başlığını kullanarak sunucuya hangi medya türlerini (veri formatlarını) desteklediğini belirtir. Örneğin, bir istemci **JSON** formatını destekliyorsa, "**Accept: application/json**" şeklinde bir başlık gönderebilir. Sunucu, **HTTP** yanıtında "**Content-Type**" başlığını kullanarak gönderilen verinin medya türünü belirtir. Örneğin, sunucu **JSON** formatında bir yanıt gönderiyorsa, "**Content-Type: application/json**" şeklinde bir başlık kullanabilir. Bu başlıklar, istemci ve sunucunun birlikte çalışabilir bir veri formatı belirleme konusunda anlaşmalarını sağlar. İstemci ve sunucu arasında desteklenen bir veya birden fazla medya türü bulunabilir ve bu konuda sunucu ve istemci arasındaki gerçekleştirilen işleme ____(d)____ denir.

API tasarımında ___(e)____, gelen verilerin belirli kurallara uygun olup olmadığını kontrol etme sürecidir. Bu süreç, veri bütünlüğünü, güvenliği ve doğruluğunu sağlamak için önemlidir. ___(e)___, **API**'lerde alınan isteklerin ve gönderilen yanıtların belirli standartlara ve kurallara uygun olup olmadığını kontrol eder.

Representational State Transfer (**REST**), or **REST** in short, is an architectural style used to communicate between web-based applications. This architectural style was introduced by Roy Fielding in 2000 as his doctoral thesis. **REST** works especially on the **HTTP** protocol and is a popular approach for creating web services. In the REST architecture, everything revolves around ___(a)__. To distinguish them from each other, Unified Resource Identifier (URI) structures, in other words, endpoints are used.

The client receives the representation of a resource (**Representational State Transfer**) from the server. This representation can generally be in **XML** or **JSON** format. The client interacts with the state of the resource based on the received representation. Each request is processed independently by the server, and each request must contain all the necessary information. The server does not store the state of the client. This concept is expressed with ___(b)____.

Clients can navigate through the information received from the server. In other words, the server provides information to the client about the state of a resource, and the client uses this information to interact with other resources. ___(c)___ is a principle used when designing a RESTful web service. This principle aims to allow clients to navigate between resources based on the representation received from the server. It fundamentally ensures that the designed API is discoverable.

In an HTTP request, it specifies to the server which media types (data formats) it supports using the "**Accept**" header. For example, if a client supports **JSON** format, it can send a header like "**Accept: application/json**." The server, in the **HTTP** response, uses the "**Content-Type**" header to specify the media type of the sent data. For instance, if the server is sending a response in **JSON** format, it can use a header like "**Content-Type: application/json**." These headers allow the client and server to come to an agreement on a compatible data format. The process carried out between the server and the client regarding the supported one or more media types is called ___(d)___.

In API design, ___(e)___ is the process of checking whether incoming data complies with certain rules. This process is crucial for ensuring data integrity, security, and accuracy. ___(e)___ checks whether the requests and responses in APIs comply with certain standards and rules.