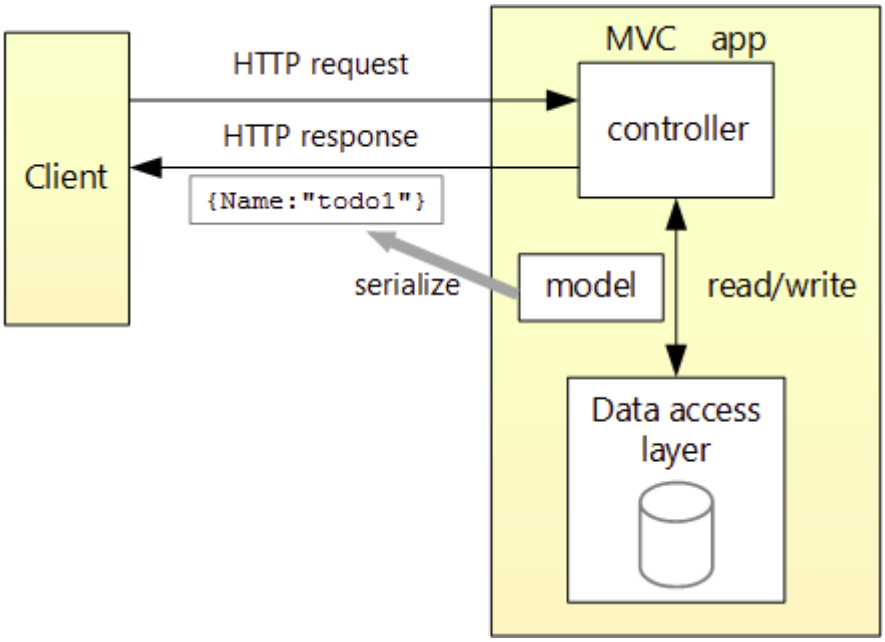
# API prenotazione esami.

## Descrizione

Creazione delle API necessarie alla prenotazione di un esame. Si presuppone che uno studente possa prenotare solo un esame alla volta.

## Architettura del progettto

Nello sviluppo delle API è stato rispettato il pattern MVC. A tal proposito ho deciso di utilizzare il l'archetipo *webapi* fornito dal framework.




## Storage

Dato il livello di disaccoppiamento fornito dal framework attraverso l'oggetto *EntityFrameworkCore* è stato possibile l'utilizzo del database in_memory. Attraverso la semplice configurazione sarà possibile sostituire il database associato con altri.

## Sviluppo del progetto

Per creare un nuovo progetto di tipo RESTFull API, si può procedere in diversi modi, in questi passi è stato utilizzato Visual Code con l'archetipo webapi.

Di seguito gli step eseguiti:

1. **Inizializzare il progetto**

   Da terminale (Visualizza / Terminale integrato) eseguire i seguenti comandi dentro la cartella del progetto

```
mkdir TodoApi
cd TodoApi
dotnet new webapi
```

> A fronte di questo comando verrà creato un template di una applicazione MVC con un *Route Template* che esporrà le nostre API nel path '/api' ed esporra' la risorsa /values.
>
> Per poter testare l'applicazione appena create, posizionarsi su Startup.cs e premere F5, ed aprendo il browser sara' possibile accedere alla risorsa values.

2. **Database *in memory***

   Per poter effettuare questo progetto e' stato utilizzato un database *in memory*, che successivamente potrà essere sostituito con altri tipi di database.

   Per includere le librerie necessarie e' stato utilizzato il package *Microsoft.EntityFrameworkCore.InMemory* nel file di configurazione del progetto.

```
  <PackageReference Include="Microsoft.EntityFrameworkCore.InMemory"
Version="2.0.1" />
```

3. **Definizione del *Model Layer* e creazione dei Data Transfer Objects.**

   Dopo aver importato le librerie per il db in memory, ho definito quale oggetti avrei voluto esporre sulle api e dunque ho definito le entità da salvare sul database rappresentate dentro il package Models.

   - Exam.cs
   - Student.cs

4. **Creazione e registrazione *db context***

   Una volta definito il modello dei dati da salvare ho creato il context che utilizzerò per memorizzare e recuperare gli oggetti che ho definito nel model.

   - Creazione classe ExamContext.cs
   - Aggiunta del contesto al services provider definito in Startup.cs

```
services.AddDbContext<ExamContext>(opt =>
opt.UseInMemoryDatabase("ExamList"));
```

5. **Definizione ExamsController**

   Alla fine ho definito il nuovo controller che dovrà esporre le mie API. Nel costruttore del controller ho inizializzato il db. Poi ho definito i seguenti metodi delle API.

   - [HttpGet] : restituisce tutti gli esami presenti
   - [HttpGet("{id}", Name = "GetExam")] : restituisce l'esame con chiave {id}
   - [HttpPost] : consente di aggiungere un nuovo esame
   - [HttpPut("{id}")] : consente di modificare un'esame esistente
   - [HttpGet("availableStudents")] : elenco di tutti gli studenti
   - [HttpGet("{id}/students")] : elenco di studenti di un determinato corso
   - [HttpPut("{id}/students")] : prenotazione esame {id}.

   > ***NOTE*** Il context-path iniziale è definito come [Route("api/[Controller]")] , dove Controller viene sostituito con exams (ovvero il nome del controller stesso).

6. **Test API**

Per effettuare i test è stato utilizzato lo strumento POSTMAN, che consente di effettuare test sulle singole chiamate, e verificare i risultati.

7. **Documentazione delle API**

   Per rendere maggiormente fruibili le API ho utilizzato il progetto swagger che rispetta le specifiche OpenAPI. Per farlo ho seguito i seguenti passi:

   - *Importazione Swashbuckle.AspNetCore*

     Ho importato i package del progetto Swashbuckle, creato appositamente per l'integrazione di swagger con la piattaforma .NET

     ```
     ```
     dotnet add TodoApi.csproj package Swashbuckle.AspNetCore
     ```
     ```

   - Configurazione

     Ho aggiunto Swagger generator nel file Startup.cs, per descrivere le API e automatizzare il processo di generazione della documentazione.

   - Test

     Una volta lanciata l'applicazione sarà possibile visulizzare la documentazione delle API direttamente dalla pagina generata da swagger, al path http://localhost:5000/swagger/. Da tale interfaccia sarà possibile testare direttamente i metodi delle API.

> **Link Utili**
>
> - microsoft documentation for webapi
> - microsoft documentation for swagger