

Backend Coding Test

TECHNICAL DESIGN DOCUMENT

Ali Hussein | 10/03/2020

- Development Tools
- API Description
- API Service Integration
- How does this API work?
- Can we handle 1,000 users? What about 1,000,000?
- How we can handle more requests? What is the bottleneck?

Development Tools

Development component choices:

- App: Visual Studio, SQL Server Management Studio (SSMS)
- Framework: .Net Framework
- Database: SQL Server
- Programming Language: C#

WHY THIS DATABASE OR FRAMEWORK

SQL Server:

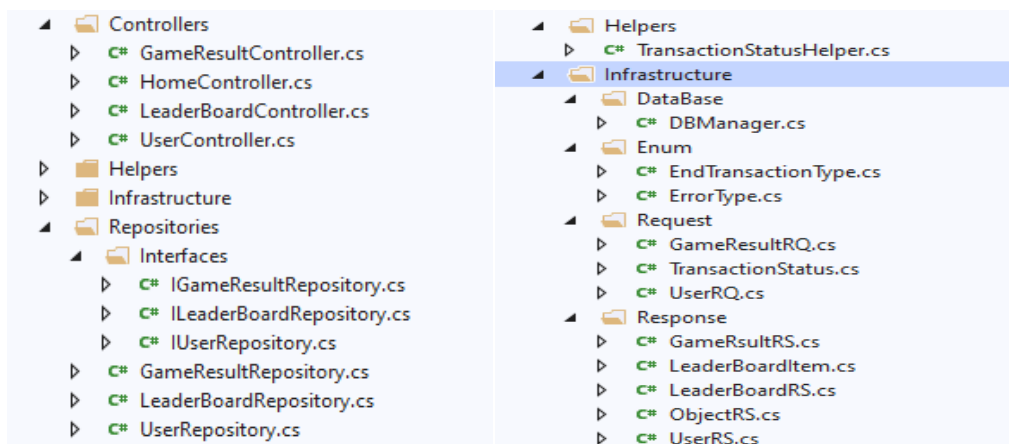
- it's a relational database management system, it supports for linux and windows platforms.
- it provides high scalability as it can be used for small projects as well as large applications. It helps in taking care of millions of transactions per day.
- It gives better performance and high speed while retrieving the data for the application.
- SQL Server has a feature to integrate with a Visual studio for data programming, SQL Server is integrating with the language .Net C#

Hint: MySQL is more popular as it is open source, free of cost and compatible with many platforms as compared to SQL server.

API Description

This API implements three Services: Register User, Report Game Result, Get LeaderBoard

1. The API Contains Three Controller : RegisterUser, GamResult, LeaderBoard.
2. Every Controller generates Request and call the repository that handles the data to implement the requirements.
3. Every Repository calls the DB Manager to impelment the required service.
4. DB Manager has the Functions that deals with DB.



REGISTER USER

UserController:

It makes call for register User Service

- Generate the Request by using the Param: UserName.
- Call the UserRepository to insert the User.
- Get the response after calling this Repository.

UserRepository:

It has the function that makes call for DBManager to Register User, by calling DBManger.instance.RegisterUser(UserRQ).

```
1 reference
public class UserRepository : IUserRepository
{
    2 references
    public async Task<bool> Add(UserRQ userName)
    {
        return await DBManager.Instance.RegisterUserAsync(userName);
    }
}
```

REPORT GAMERESULT:

GameResultCotroller:

- Generate the Request by using the Param: UserName, Score.
- Call GameResultRepository to report the score.
- Get the response after calling this Repository.

GameResult Repository:

It makes the process to Report GameResult by calling DBManager to report the Score

- first check the User if it's registered by calling: DBManger.instance.CheckUserRegistered(UserName)
- if the User is Registered then insert his Score by calling: DBmanger.instance.InsertUserScore(IdUser, Score)

LEADERBOARD:

LeaderBoard Controller:

- Generate the Request by using the Param: TopUsersNumber
- Call LeaderBoardRepository to get the LeaderBoard.
- Get the response after calling this Repository.

LeaderBoard Repository

it has the function that get leaderboard of top scores.

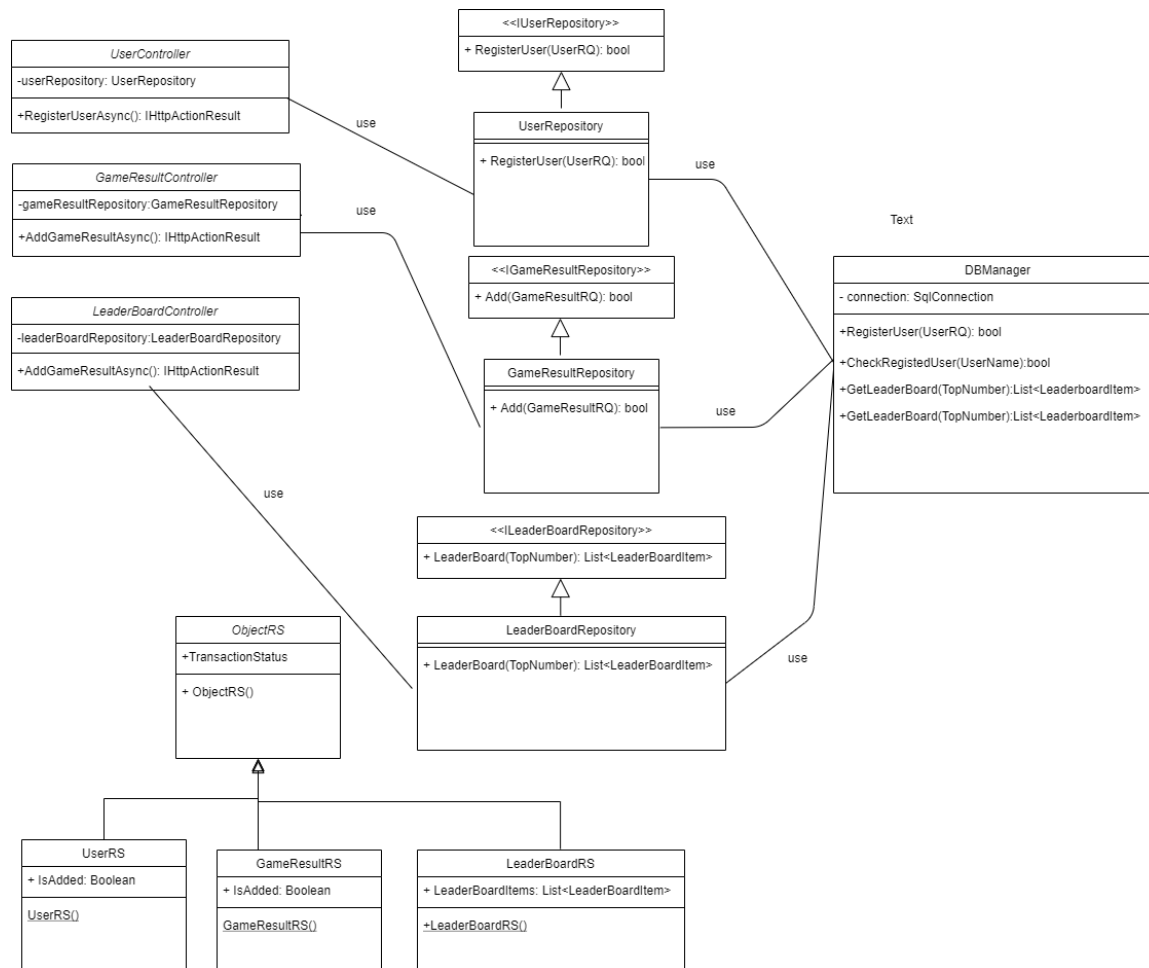
by calling `DBManger.instance.GetLeaderBoard(TopUsersNumber)`.

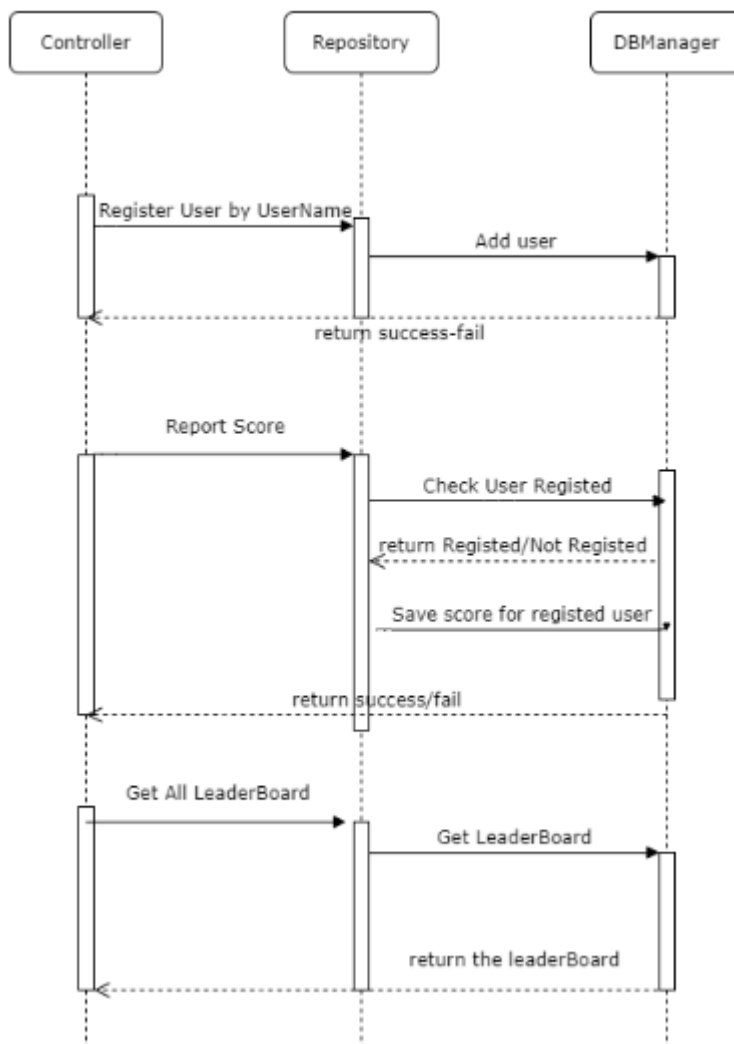
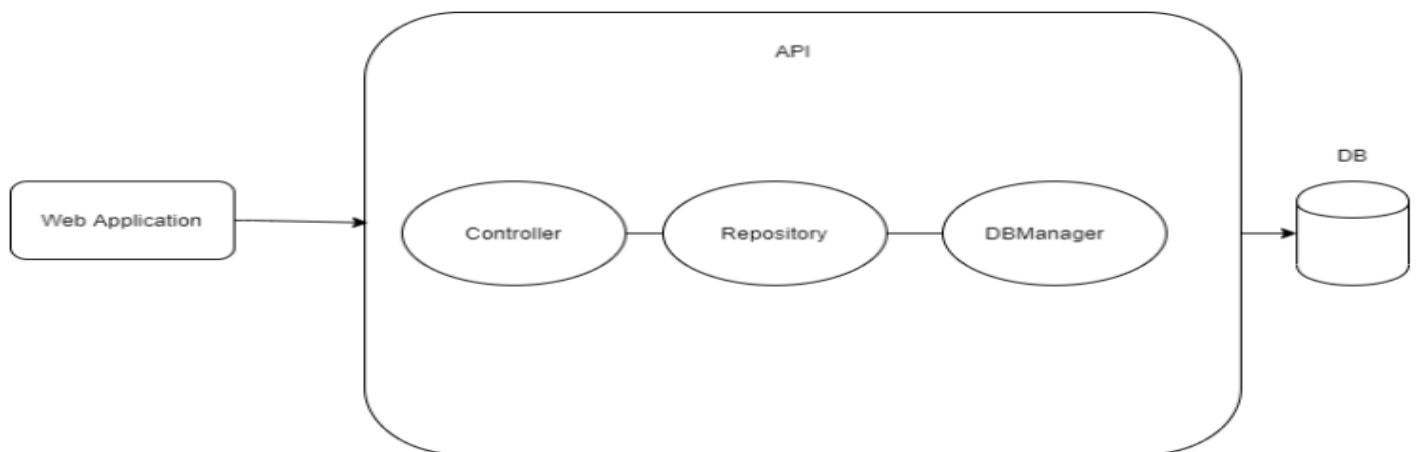
DBMANAGER:

it's a singleton class that has the connection with DB to insert and the data.

- Every Function get the Params and StoreProcedure name and execute it.
- Functions: RegisterUser, CheckUserRegistered , ReportGameResult, GetLeaderBoard.

UML Diagram

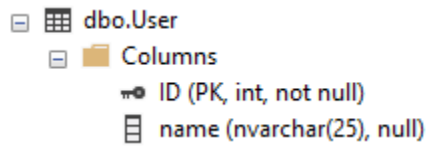




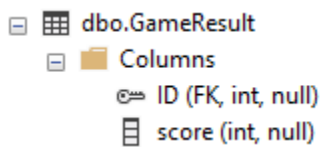
DataBase

it has Two Tables:

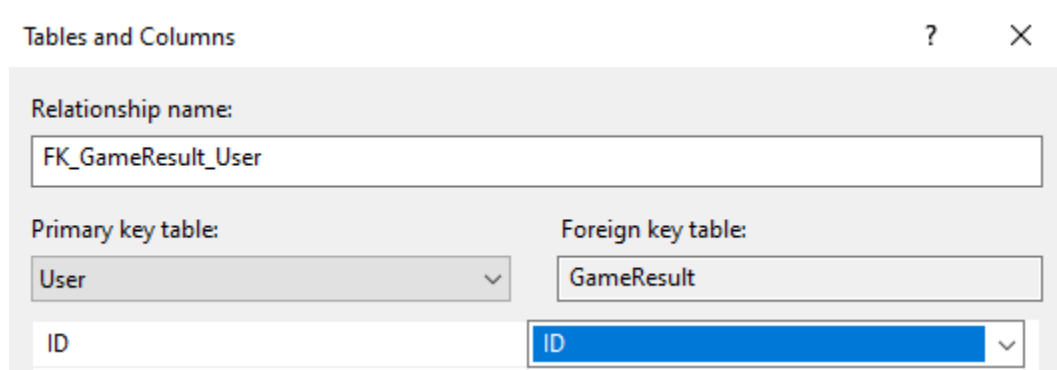
User Table: to register the user Param



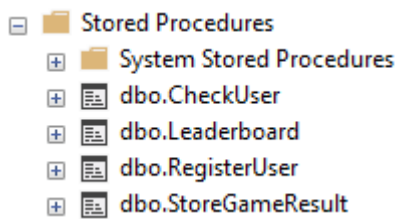
GameResult Table: to report the score



Relational Between User , GameResult Table



it has four storedProcedures



RegisterUser Procedure:

```
USE [BackendCodingTest]
GO
/***** Object:  StoredProcedure [dbo].[RegisterUser]    Sc
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[RegisterUser](@name nvarchar(25))
as
Insert into [User] (name) values (@name)
return
```

CheckUserRegistered:

```
USE [BackendCodingTest]
GO
/***** Object:  StoredProcedure [dbo].[CheckUser]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Procedure [dbo].[CheckUser](@name nvarchar(25))
As
Begin
Select [ID], [name]
from [User]
where [name] = @name
End
```

AddGameResult:

```
USE [BackendCodingTest]
GO
/***** Object:  StoredProcedure [dbo].[StoreGameResult]    Sc
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Procedure [dbo].[StoreGameResult](@id int, @score int)
As
Begin
Insert into [GameResult] (ID, score)
values (@id, @score)
End
```

GetLeaderBoard:

```
USE [BackendCodingTest]
GO
/***** Object:  StoredProcedure [dbo].[Leaderboard]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Procedure [dbo].[Leaderboard]
As
Begin
Select [User].name, [GameResult].score
From [User] inner join [GameResult]
On [User].ID = [GameResult].ID
Order by [GameResult].score DESC
End
```


How does it work

First Run the Project from Visual Studio

Then open this url path: <https://localhost:44356/swagger/ui/index>

we have three Functions

- Register User with Post Method
- Report Score with Post Method for first time, if it's second time it should be Update Method
- GetLeaderBoard by using Get Method.

 **swagger**

Explore

Codingtest.Chimera.Api

GameResult

Show/Hide | List Operations | Expand Operations

POST /GameResult/AddScore

LeaderBoard

Show/Hide | List Operations | Expand Operations

GET /GameResult/GetLeaderBoard

User

Show/Hide | List Operations | Expand Operations

POST /user/registerUser

[BASE URL: , API VERSION: v1]

RegisterUser:

User

Show/Hide | List Operations | Expand Operations

POST /user/registerUser

Response Class (Status 200)

string

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
name	<input type="text" value="Ali"/>		query	string

Try it out! [Hide Response](#)

Curl

curl -X POST --header 'Accept: application/json' 'https://localhost:44356/user/registerUser?name=Ali'

Request URL

https://localhost:44356/user/registerUser?name=Ali

Response Body

```
{
  "IsAdded": true,
  "TransactionStatus": {
    "Code": "200",
    "SubCode": null,
    "Message": null,
    "Duration": null,
    "ErrorType": 0,
    "EndTransactionType": 0
  }
}
```


Report Score:

POST /GameResult/AddScore

Response Class (Status 200)
string

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
name	<input type="text" value="Ali"/>		query	string
score	<input type="text" value="200"/>		query	integer

[Try it out!](#) [Hide Response](#)

Curl

```
curl -X POST --header 'Accept: application/json' 'https://localhost:44356/GameResult/AddScore?name=Ali&score=200'
```

Request URL

```
https://localhost:44356/GameResult/AddScore?name=Ali&score=200
```

Response Body

```
{
  "IsAdded": true,
  "TransactionStatus": {
    "Code": "200",
    "SubCode": null,
    "Message": null,
    "Duration": null,
    "ErrorType": 0,
    "EndTransactionType": 0
  }
}
```

GetLeaderBoard:

LeaderBoard

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

GET /GameResult/GetLeaderBoard

Response Class (Status 200)
string

Response Content Type

[Try it out!](#) [Hide Response](#)

Curl

```
curl -X GET --header 'Accept: application/json' 'https://localhost:44356/GameResult/GetLeaderBoard'
```

Request URL

```
https://localhost:44356/GameResult/GetLeaderBoard
```

Response Body

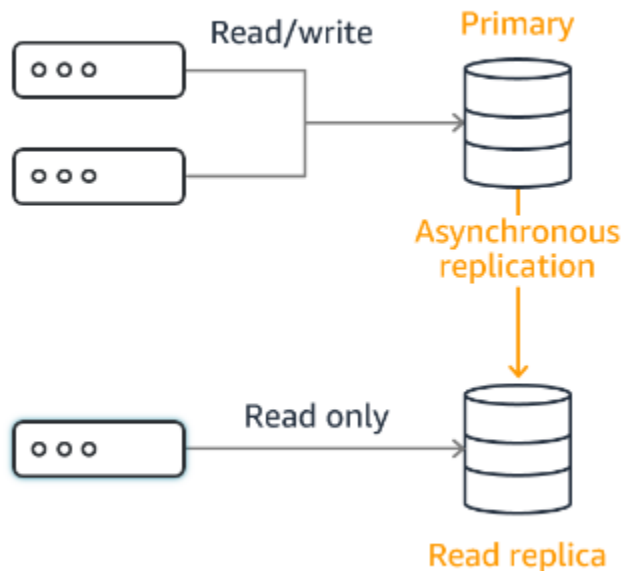
```
{
  "leaderBoardItems": [
    {
      "UserName": "Hussein",
      "Score": 70
    },
    {
      "UserName": "Ali",
      "Score": 50
    }
  ],
  "TransactionStatus": {
    "Code": "200",
    "SubCode": null,
    "Message": null,
    "Duration": null,
    "ErrorType": 0,
    "EndTransactionType": 0
  }
}
```

Can we handle 1,000 users? What about 1,000,000?

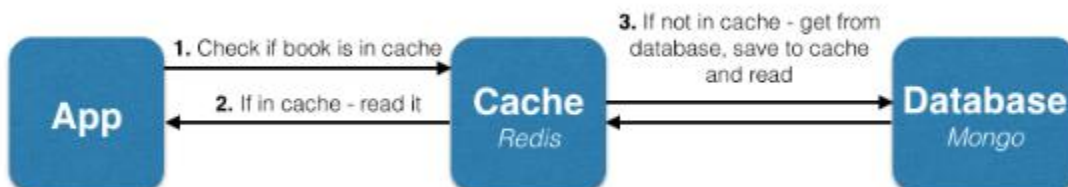
1,000 it depends on DB storage. So to solve the problem of handling 1,000,000 in DB use Read Replica:

- by reading the data from read replica (multiple copies of the database) to reduce the load on main DB.
- Sync the data that in main database with the read replica (has cached data) every 10 seconds for example.
- so at the end Read Replica will be eventually consistent with the main database.

Application servers Database server



- Use Redis to cache all the query results from the database. to reduce load and times of hitting on db.



How we can handle more requests? What is the bottleneck?

- Use Async Code: if many concurrent Request in parallel we need to use asynchronous code to provide scalability.
- Use Load balancer: load balancing system gives you the option to distribute incoming traffic to several servers.
- Improving scalability by using more servers and scale them in two ways, horizontal and (vertical Horizontal scaling, Vertical scaling).
- Increase the resources available to each server: More CPUs, more RAM, more disk storage, faster internet, increase the number of threads.

What is the bottleneck?

It is the slowest point in the system or the rarest resource.

If you have multiple CPUs, multiple threads but one database for example, all the different threads will compete to talk to the database and each thread will have to wait for some time in order to be able to connect to the database, in that case the database will be the bottleneck.