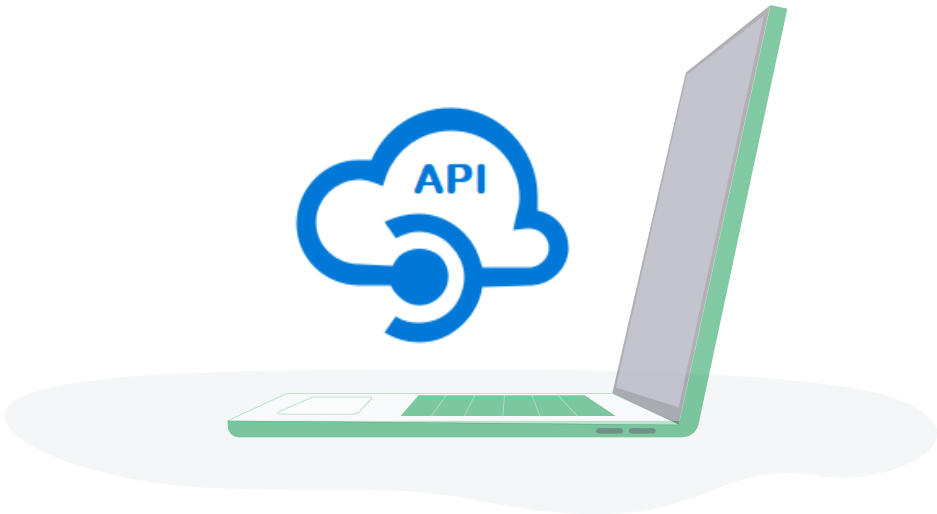
Web Application Programming Interface (API)

Tahaluf Training Center 2023

1. Overview of CRUD Operations
2. Repository



Overview of CRUD Operations

CRUD is an acronym that comes from the computer programming world. It refers to the four functions that are represented necessary to implement a persistent storage application.

CRUD: Create, Read, Update and **Delete**.

Create

The create function allows users to create a new row in the database.

In the SQL relational database, the Create function is called INSERT.

Read

The read function is a search function. It allows users to retrieve and search specific rows in the table and read their values.

In the SQL relational database, the Read function is called SELECT.

Update

The update function is used to update existing rows that exist in the database. To fully change a record, users may have to update information in multiple fields.

In the SQL relational database, the Update function is called UPDATE.

Delete

The delete function allows users to delete rows from a database that is no longer needed. Both Oracle HCM Cloud and SQL have a delete function that allows users to delete one or more rows from the database.

In the SQL relational database, the Delete function is called DELETE.

Repository

Repository Layer

A repository Layer is intended to build an abstraction layer between the business logic layer and the domain layer of an application. It is a domain approach that prompts a more loosely coupled pattern to data access.

- Right Click on LearningHub.Infra => Add => New Folder => Repository.
- Right Click on LearningHub.Core => Add => New Folder => Repository.
- Right Click on Repository Folder in LearningHub.Core => Add => Class => Interface => ICourseRepository.
- Right Click on Repository Folder in LearningHub.Infra => Add => Class => CourseRepository.

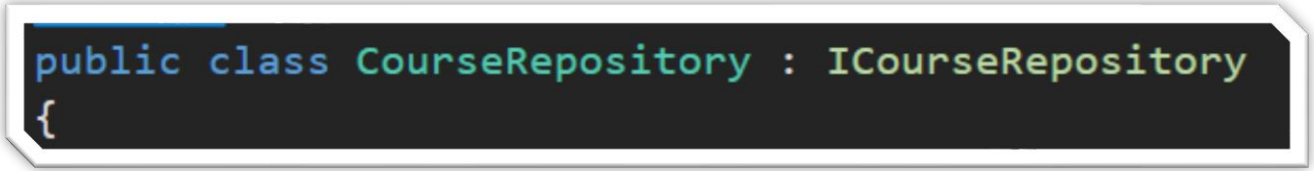
- **Note:**
- Make sure all created classes and interfaces are public.

In LearningHub.Core => Repository => ICourseRepository add the following abstract methods:

```
List<Course> GetAllCourse();
void CreateCourse(Course course);
void DeleteCourse(int id);
public void UpdateCourse(Course course);
Course GetByCourseId(int id);
```

In LearningHub.Infra => Repository => CourseRepository => make the class inherit the interface ICourseRepository:

```
public class CourseRepository : ICourseRepository
```



In LearningHub.Infra => Repository => CourseRepository add the following :

```
private readonly IDBContext dBContext;

public CourseRepository(IDBContext dBContext)
{
    this.dBContext = dBContext;
}}
```

In LearningHub.Infra => Repository => CourseRepository add the following :

```
public List<Course> GetAllCourse()
            {
            IEnumerable<Course> result =
dBContext.Connection.Query<Course>("Course_Package.GetA
llCourses", commandType: CommandType.StoredProcedure);
            return result.ToList();
            }
```

In LearningHub.Infra => Repository => CourseRepository add the following :

```
public void CreateCourse(Course course)
        {
                var p = new DynamicParameters();
                p.Add("COURSENAME", course.Coursename,
dbType: DbType.String, direction:
ParameterDirection.Input);
                p.Add("CATID", course.Categoryid,
dbType: DbType.Int32, direction:
ParameterDirection.Input);
```

```
p.Add("image", course. Imagename, dbType:
DbType.String, direction: ParameterDirection.Input);

            var result =
dBContext.Connection.Execute("Course_Package.CREATECOUR
SE", p, commandType: CommandType.StoredProcedure);

        }
```

In LearningHub.Infra => Repository => CourseRepository add the following :

```
 public void UpdateCourse(Course course)
            {
var p = new DynamicParameters();
            p.Add("ID", course.Courseid, dbType:
DbType.Int32, direction: ParameterDirection.Input);
            p.Add("CNAME", course.Coursename, dbType:
DbType.String, direction: ParameterDirection.Input);
```

```
p.Add("CATID", course.Categoryid, dbType:
DbType.Int32, direction: ParameterDirection.Input);
p.Add("image", course. Imagename, dbType:
DbType.String, direction: ParameterDirection.Input);
            var result =
dBContext.Connection.Execute("Course_Package.UPDATECO
URSE", p, commandType: CommandType.StoredProcedure);
            }
```

In LearningHub.Infra => Repository => CourseRepository add the following :

```
public void DeleteCourse(int id)
        {
            var p = new DynamicParameters();
            p.Add("Id", id, dbType: DbType.Int32,
direction: ParameterDirection.Input);
            var result =
dBContext.Connection.Execute("Course_Package.DeleteCour
se", p, commandType: CommandType.StoredProcedure);
        }
```

In LearningHub.Infra => Repository => CourseRepository add the following :

```
            public Course GetByCourseId(int id)
             {
             var p = new DynamicParameters();
             p.Add("id", id, dbType: DbType.Int32,
direction: ParameterDirection.Input);
             IEnumerable<Course> result =
dBContext.Connection.Query<Course>("Course_Package.GetC
ourseById", p, commandType:
CommandType.StoredProcedure);
             return result.FirstOrDefault();
             }
```

Add Services in Program

Write the following code in Configure services:

```
builder.Services.AddScoped<ICourseRepository, CourseRepository>();
```

- Right Click on Repository Folder in LearningHub.Core => Add => Class => Interface => IStudentRepository.
- Right Click on Repository Folder in LearningHub.Infra => Add => Class => StudentRepository.

Note:

Make sure all created classes and interfaces are public.

In LearningHub.Core => Repository => IStudentRepository add the following abstract methods:

```
List<Student> GetAllStudent();
void CreateStudent(Student Student);
void UpdateStudent(Student Student);
void DeleteStudent(int id);
Student GetStudentById(int id);
```

In LearningHub.Infra => Repository => StudentRepository => make the class inherit the interface IStudentRepository:

```
public class StudentRepository : IStudentRepository
```

```
public class StudentRepository : IStudentRepository
{
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
private readonly IDBContext dBContext;

        public StudentRepository(IDBContext dBContext)
        {
            this.dBContext = dBContext;
        }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
public List<Student> GetAllStudent()
            {
            IEnumerable<Student> result =
dBContext.Connection.Query<Student>("Student_Package.Ge
tAllStudent", commandType:
CommandType.StoredProcedure);
            return result.ToList();
            }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
        public void CreateStudent(Student Student)
            {
            var p = new DynamicParameters();
            p.Add("first_name", Student.Firstname,
dbType: DbType.String, direction:
ParameterDirection.Input);
            p.Add("last_name", Student.Lastname,
dbType: DbType.String, direction:
ParameterDirection.Input);
```

```
p.Add("date_of_birth", Student.Dateofbirth, dbType:
DbType.DateTime, direction: ParameterDirection.Input);
            var result =
dBContext.Connection.ExecuteAsync("Student_Package.Cre
ateStudent", p, commandType:
CommandType.StoredProcedure);
            }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
 public void UpdateStudent(Student Student)
            {
            var p = new DynamicParameters();
            p.Add("ID", Student.Studentid, dbType:
DbType.Int32, direction: ParameterDirection.Input);
            p.Add("first_name", Student.Firstname,
dbType: DbType.String, direction:
ParameterDirection.Input);
            p.Add("last_name", Student.Lastname,
dbType: DbType.String, direction:
ParameterDirection.Input);
```

```
p.Add("date_of_birth", Student.Dateofbirth, dbType:
DbType.DateTime, direction: ParameterDirection.Input);
            var result =
dBContext.Connection.ExecuteAsync("Student_Package.Upda
teStudent", p, commandType:
CommandType.StoredProcedure);
            }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
 public void DeleteStudent(int id)
            {
            var p = new DynamicParameters();
            p.Add("ID", id, dbType: DbType.Int32,
direction: ParameterDirection.Input);
            var result =
dBContext.Connection.ExecuteAsync("Student_Package.Dele
teStudent", p, commandType:
CommandType.StoredProcedure);
            }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
 public Student GetStudentById(int id)
            {
            var p = new DynamicParameters();
            p.Add("ID", id, dbType: DbType.Int32,
direction: ParameterDirection.Input);
            IEnumerable<Student> result =
dBContext.Connection.Query<Student>("Student_Package.Ge
tStudentById", p, commandType:
CommandType.StoredProcedure);
            return result.FirstOrDefault();
            }
```

Add Services in Program

Write the following code in Configure services:

```
builder.Services.AddScoped<IStudentRepository, StudentRepository>();
```

- Right Click on Repository Folder in LearningHub.Core => Add => Class => Interface => IStudentCourseRepository .

- Right Click on Repository Folder in LearningHub.Infra => Add => Class => StudentCourseRepository.

Note:

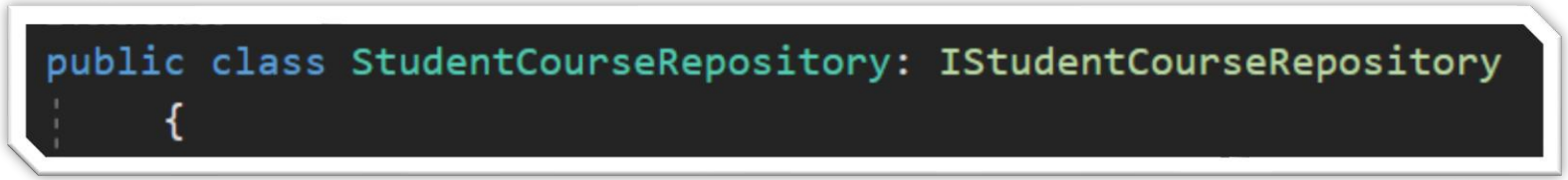
Make sure all created classes and interfaces are public.

In LearningHub.Core => Repository => IStudentCourseRepository add the following abstract methods:

```
 List<Stdcourse> GetAllStudentCourse();
        void CreateStudentCourse(Stdcourse
studentCourse);
        void DeleteStudentCourse(int id);
        void UpdateStudentCourse(Stdcourse
studentCourse);
        Stdcourse GetStudentCourseById(int id);
```

In LearningHub.Infra => Repository => StudentCourseRepository => make the class inherit the interface IStudentCourseRepository:

```
public class StudentCourseRepository: IStudentCourseRepository
```



In LearningHub.Infra => Repository => StudentCourseRepository add the following :

```
 private readonly IDBContext dBContext;

        public StudentCourseRepository(IDBContext
dBContext)
            {
            this.dBContext = dBContext;
            }
```

In LearningHub.Infra => Repository => StudentCourseRepository add the following :

```
 public void CreateStudentCourse(StdCourse
studentCourse)
            {
            var p = new DynamicParameters();
            p.Add("stdidid", studentCourse.Stdid,
dbType: DbType.Int32, direction:
ParameterDirection.Input);
            p.Add("courseid", studentCourse.Courseid,
dbType: DbType.Int32, direction:
ParameterDirection.Input);
```

In LearningHub.Infra => Repository => StudentCourseRepository add the following :

```
p.Add("markof", studentCourse.Markofstd, dbType:
DbType.Int32, direction: ParameterDirection.Input);
            p.Add("dateof_register",
studentCourse.Dateofregister, dbType: DbType.DateTime,
direction: ParameterDirection.Input);
            var result =
dBContext.Connection.ExecuteAsync("stdcourse_Package.Cr
eateStdCourse", p, commandType:
CommandType.StoredProcedure);
            }
```

In LearningHub.Infra => Repository => StudentCourseRepository add the following :

```
 public void DeleteStudentCourse(int id)
            {
            var p = new DynamicParameters();
            p.Add("SCID", id, dbType: DbType.Int32,
direction: ParameterDirection.Input);
            var result =
dBContext.Connection.ExecuteAsync("stdcourse_Package.De
leteStdCourse", p, commandType:
CommandType.StoredProcedure);
            }
```

In LearningHub.Infra => Repository => StudentCourseRepository add the following :

```
public List<StdCourse> GetAllStudentCourse()
            {
            IEnumerable<StdCourse> result =
dBContext.Connection.Query<StdCourse>("stdcourse_Packag
e.GetAllStdCourse", commandType:
CommandType.StoredProcedure);
            return result.ToList();
            }
```

In LearningHub.Infra => Repository => StudentCourseRepository add the following :

```
  public StdCourse GetStudentCourseById(int id)
            {
            var p = new DynamicParameters();
            p.Add("SCID", id, dbType: DbType.Int32,
direction: ParameterDirection.Input);
            IEnumerable<StdCourse> result =
dBContext.Connection.Query<StdCourse>("stdcourse_Packag
e.GetStdCourseById", p, commandType:
CommandType.StoredProcedure);
            return result.FirstOrDefault();
            }
```

In LearningHub.Infra => Repository => StudentCourseRepository add the following :

```
public void UpdateStudentCourse(StdCourse studentCourse)
            {
            var p = new DynamicParameters();
            p.Add("SCid", studentCourse.Id, dbType:
DbType.Int32, direction: ParameterDirection.Input);
            p.Add("stdidid", studentCourse.Stdid, dbType:
DbType.Int32, direction: ParameterDirection.Input);
            p.Add("courseid", studentCourse.Courseid, dbType:
DbType.Int32, direction: ParameterDirection.Input);
```

```
p.Add("markof", studentCourse.Markofstd, dbType:
DbType.Int32, direction: ParameterDirection.Input);
            p.Add("dateof_register",
studentCourse.Dateofregister, dbType: DbType.DateTime,
direction: ParameterDirection.Input);
            var result =
dBContext.Connection.ExecuteAsync("stdcourse_Package.UpdateSt
dCourse", p, commandType: CommandType.StoredProcedure);
            }
```

Add Services in Program

Write the following code in Configure services:

```
builder.Services.AddScoped<IStudentCourseRepository, StudentCourseRepository>();
```

Exercise

- ✓ Create a function to display FirstName and LastName from table student.
- ✓ Create a function to display students by firstName.
- ✓ Create a function to display students by BirthOfDate.
- ✓ Create a function to display a student by BirthOfDate interval.
- ✓ Create a function to display the student name with the highest n(2,3,…) marks

In LearningHub.Core => Repository => IStudentRepository add the following :

```
 List<Student> GetStudentByFName(string name);
        List<Student> GetStudentFNameAndLName();
        List<Student> GetStudentByBirthdate(DateTime
Birth_Date);
        List<Student> GetStudentBetweenDate(DateTime
DateFrom  ,DateTime DateTo );
        List<Student> GetStudentsWithHighestMarks(int
numOfStudent);
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
public List<Student> GetStudentByFName(string name)
            {
            var p = new DynamicParameters();
            p.Add("First_Name", name, dbType:
DbType.String, direction: ParameterDirection.Input);
            IEnumerable<Student> result =
dBContext.Connection.Query<Student>("Student_Package.Ge
tStudentByFirstName", p, commandType:
CommandType.StoredProcedure);
            return result.ToList();
            }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
public List<Student> GetStudentFNameAndLName()
            {
            IEnumerable<Student> result =
dBContext.Connection.Query<Student>("Student_Package.Ge
tStudentFNameAndLName", commandType:
CommandType.StoredProcedure);
            return result.ToList();
            }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
 public List<Student> GetStudentByBirthdate(DateTime
Birth_Date)
            {
            var p = new DynamicParameters();
            p.Add("Birth_Date", Birth_Date, dbType:
DbType.DateTime, direction: ParameterDirection.Input);
            IEnumerable<Student> result =
dBContext.Connection.Query<Student>("Student_Package.GetStudent
ByBirthdate", p, commandType: CommandType.StoredProcedure);
            return result.ToList();
            }
```

In LearningHub.Infra => Repository => StudentRepository add the following :

```
 public List<Student> GetStudentBetweenDate(DateTime DateFrom,
DateTime DateTo)
            {
            var p = new DynamicParameters();
            p.Add("DateFrom", DateFrom, dbType:
DbType.DateTime, direction: ParameterDirection.Input);
            p.Add("DateTo", DateTo, dbType: DbType.DateTime,
direction: ParameterDirection.Input);
            IEnumerable<Student> result =
dBContext.Connection.Query<Student>("Student_Package.GetStudent
BetweenInterval", p, commandType: CommandType.StoredProcedure);
            return result.ToList();
            }
```

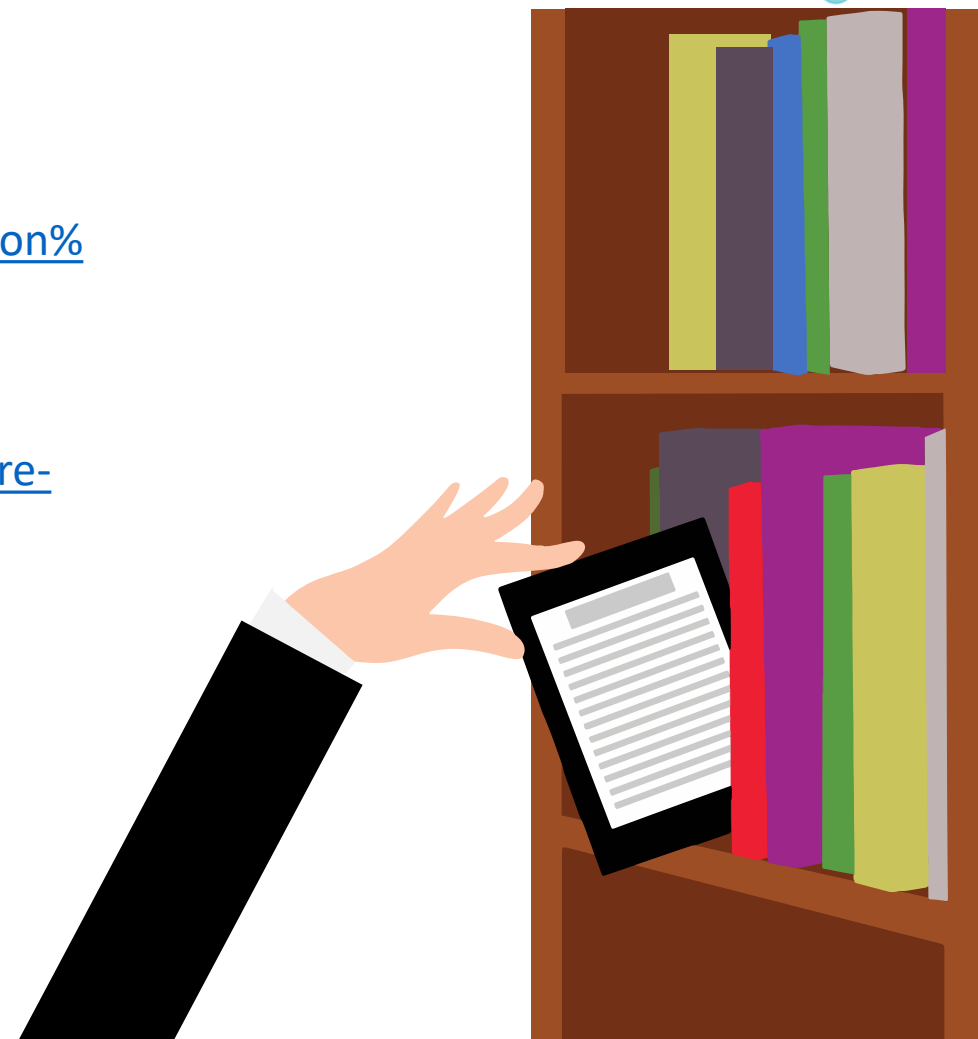
In LearningHub.Infra => Repository => StudentRepository add the following:

```
 public List<Student> GetStudentsWithHighestMarks(int
numOfStudent)
            {
            var p = new DynamicParameters();
            p.Add("NumOfStudent", numOfStudent, dbType:
DbType.Int32, direction: ParameterDirection.Input);
            IEnumerable<Student> result =
dBContext.Connection.Query<Student>("Student_Package.GetStudent
sWithHighestMarks",p, commandType:
CommandType.StoredProcedure);
            return result.ToList();
            }
```

References

[1]. https://www.codeguru.com/csharp/understanding-onion-architecture/#:~:text=Onion%20Architecture%20is%20based%20on,on%20the%20actual%20domain%20models

[2]. https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-5.0



The Learning Hub