

ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



BLM4531 - Ağ Tabanlı Teknolojiler ve Uygulamaları

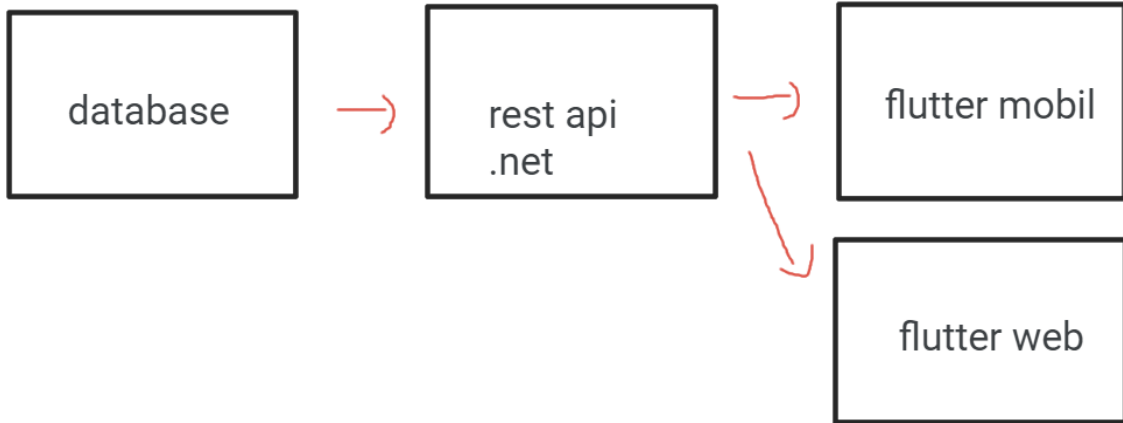
**Proje Raporu
OBS Uygulaması**

Refik Can ÖZTAŞ

19290266

12/01/2023

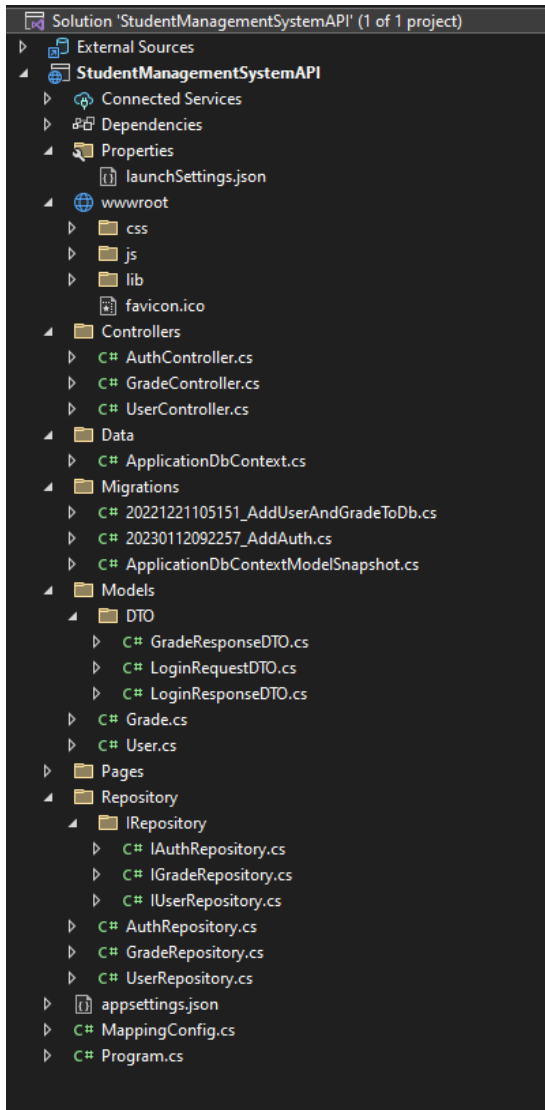
BLM4531 için bir OBS uygulaması yazılmıştır. Web ve mobil uygulamanın çalışma pipeline'ı kabaca aşağıdaki gibidir:



web uygulaması : <https://github.com/canoztas/StudentManagementSystemWeb>
api : <https://github.com/canoztas/StudentManagementSystemAPI>

Backend Kod Kısmı:

API kısmında:



Şeklinde bir yapı kurulmuştur.

Modeller iki tanedir:

```
namespace StudentManagementSystemAPI.Models
{
    26 references
    public class User
    {
        8 references
        public int UserId { get; set; }
        4 references
        public string FirstName { get; set; }
        4 references
        public string LastName { get; set; }
        7 references
        public string Email { get; set; }
        10 references
        public string PasswordHash { get; set; }
        4 references
        public string PhotoPath { get; set; }
        5 references
        public string UserType { get; set; }
    }
}
```

```
namespace StudentManagementSystemAPI.Models
{
    25 references
    public class Grade
    {
        5 references
        public int GradeId { get; set; }
        5 references
        public string LessonName { get; set; }
        6 references
        public int LecturerId { get; set; }
        12 references
        public int StudentId { get; set; }
        9 references
        public int Score { get; set; }
        9 references
        public string LessonDate { get; set; }
    }
}
```

İki modelin de IRepository'si ve Repository'leri vardır:

```
using StudentManagementSystemAPI.Models;

namespace StudentManagementSystemAPI.Repository.IRepository
{
    4 references
    public interface IUserRepository
    {
        2 references
        Task<User> GetAsync(int id);
        2 references
        Task CreateAsync(User user);
        2 references
        Task UpdateAsync(User user);
        2 references
        Task RemoveAsync(User user);
        1 reference
        Task SaveAsync();
    }
}
```

```
namespace StudentManagementSystemAPI.Repository
{
    2 references
    public class UserRepository : IUserRepository
    {
        private readonly ApplicationDbContext _context;

        0 references
        public UserRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        2 references
        public async Task<User> GetAsync(int id)
        {
            return await _context.Users.FindAsync(id);
        }

        2 references
        public async Task CreateAsync(User user)
        {
            await _context.Users.AddAsync(user);
        }

        2 references
        public async Task UpdateAsync(User user)
        {
            _context.Users.Update(user);
        }

        2 references
        public async Task RemoveAsync(User user)
        {
            _context.Users.Remove(user);
        }
    }
}
```

```
using Microsoft.EntityFrameworkCore;
using StudentManagementSystemAPI.Models;
using StudentManagementSystemAPI.Models.DTO;

namespace StudentManagementSystemAPI.Repository.IRepository
{
    4 references
    public interface IGradeRepository
    {
        2 references
        Task<GradeResponseDTO> GetAsync(int id);
        2 references
        Task CreateAsync(Grade grade);
        2 references
        Task UpdateAsync(Grade grade);
        2 references
        Task RemoveAsync(Grade grade);
        2 references
        List<GradeResponseDTO> GetByStudentId(int id);
        2 references
        List<GradeResponseDTO> GetByLecturerId(int id);
        1 reference
        Task SaveAsync();
    }
}
```

```
namespace StudentManagementSystemAPI.Repository
{
    2 references
    public class GradeRepository : IGradeRepository
    {
        private readonly ApplicationDbContext _context;

        0 references
        public GradeRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        2 references
        public async Task<GradeResponseDTO> GetAsync(int id)
        {
            Grade grade = new Grade();
            grade = await _context.Grades.FindAsync(id);
            List<User> query = _context.Users.ToList();
            GradeResponseDTO gradeResponseDTO = new GradeResponseDTO()
            {
                Grade = grade,
                Lecturer = query.SingleOrDefault(g => g.UserId == grade.LecturerId),
                Student = query.SingleOrDefault(g => g.UserId == grade.StudentId)
            };
            gradeResponseDTO.Student.PasswordHash = null;
            gradeResponseDTO.Lecturer.PasswordHash = null;
            return gradeResponseDTO;
        }
    }
}
```

```

12 references
public class ApplicationDbContext : DbContext
{
    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
        Database.EnsureDeleted();
        Database.EnsureCreated();
    }

    8 references
    public DbSet<User> Users { get; set; }
    6 references
    public DbSet<Grade> Grades { get; set; }
}

```

Routing ise Controllerlar ile sağlanmıştır:

```

Namespace StudentManagementSystemAPI.Controllers
{
    [Route("api/[controller]")]
    1 reference
    public class GradeController : Controller
    {
        private readonly IGradeRepository _gradeRepository;

        0 references
        public GradeController(IGradeRepository gradeRepository)
        {
            _gradeRepository = gradeRepository;
        }

        [HttpGet]
        0 references
        public async Task<ActionResult<Grade>> Get(int? gradeId, int? lecturerId, int? studentId)
        {
            if (gradeId != null)
            {
                var grade = await _gradeRepository.GetAsync(gradeId.Value);
                return Ok(grade);
            }

            if (studentId != null)
            {
                var grade = _gradeRepository.GetByStudentId(studentId.Value);
                return Ok(grade);
            }

            if (lecturerId != null)
            {
                var grade = _gradeRepository.GetByLecturerId(lecturerId.Value);
                return Ok(grade);
            }

            return BadRequest();
        }
    }
}

```

```

[Route("api/[controller]")]
1 reference
public class UserController : Controller
{
    private readonly IUserRepository _userRepository;

    0 references
    public UserController(IUserRepository userRepository)
    {
        _userRepository = userRepository;
    }

    [HttpGet]
    0 references
    public async Task<ActionResult<User>> Get(int id)
    {
        var user = await _userRepository.GetAsync(id);
        return Ok(user);
    }

    [HttpPost]
    0 references
    public async Task<ActionResult> Create(User user)
    {
        await _userRepository.CreateAsync(user);
        return Ok();
    }

    [HttpPut]
    0 references
    public async Task<ActionResult> Update(User user)
    {
        await _userRepository.UpdateAsync(user);
        return Ok();
    }
}

```

Uygulamanın önemli bir kısmını oluşturan yetkilendirme için jwt kullanılmıştır.

```
4 references
public async Task<LoginResponseDTO> Login(LoginRequestDTO loginRequestDTO)
{
    var user = _db.Users.SingleOrDefault(x => x.Email == loginRequestDTO.Email && x.PasswordHash == loginRequestDTO.Password);
    //bool isValid = await _userManager.CheckPasswordAsync(user, loginRequestDTO.Password);

    //user not found
    if (user == null)
    {
        return null;
    }

    //var roles = await _userManager.GetRolesAsync(user);

    var tokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(secretKey);
    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(new Claim[]
        {
            new Claim(ClaimTypes.Email, user.Email),
            new Claim(ClaimTypes.Role, user.UserType),
        }),
        Expires = DateTime.UtcNow.AddDays(7),
        SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
    };

    var token = tokenHandler.CreateToken(tokenDescriptor);
    LoginResponseDTO loginResponseDTO = new()
    {
        User = _mapper.Map<User>(user),
        Token = new JwtSecurityTokenHandler().WriteToken(token)
    };
    loginResponseDTO.User.PasswordHash = null;
    return loginResponseDTO;
}
```

jwt için AuthRepository'si oluşturulmuştur. Bu repository User objesinin rolüne göre yetkilendirme sağlayıp jwt döner. Bu repository AuthController ile çağrılır.

```
4 references
public interface IAuthRepository
{
    2 references
    Task<LoginResponseDTO> Login(LoginRequestDTO loginRequestDTO);
}
```

```
1 reference
[Route("api/login")]
public class AuthController : Controller
{
    private readonly IAuthRepository _authRepository;

    0 references
    public AuthController(IAuthRepository authRepository)
    {
        _authRepository = authRepository;
    }

    [HttpPost]
    [HttpOptions]
    [DisableCors]
    0 references
    public async Task<ActionResult> Login([FromBody] LoginRequestDTO model)
    {
        var loginResponse = await _authRepository.Login(model);
        if (loginResponse == null)
        {
            return BadRequest();
        }
        return Ok(loginResponse);
    }
}
```

Bu repository için Request ve Response olarak iki adet DTO oluşturulmuştur. Bu objeler aracılığı ile e-mail pass alınıp, jwt döner.

```
namespace StudentManagementSystemAPI.Models.DTO
{
    4 references
    public class LoginRequestDTO
    {
        1 reference
        public string Email { get; set; }
        1 reference
        public string Password { get; set; }
    }
}
```

```
namespace StudentManagementSystemAPI.Models.DTO
{
    5 references
    public class LoginResponseDTO
    {
        2 references
        public User User { get; set; }

        1 reference
        public string Token { get; set; }
    }
}
```

Api endpointlerindeki en önemli sorgu kısmı grade endpointidir.

```
2 references
public async Task<GradeResponseDTO> GetAsync(int id)
{
    Grade grade = new Grade();
    grade = await _context.Grades.FindAsync(id);
    List<User> query = _context.Users.ToList();
    GradeResponseDTO gradeResponseDTO = new GradeResponseDTO()
    {
        Grade = grade,
        Lecturer = query.SingleOrDefault(g => g.UserId == grade.LecturerId),
        Student = query.SingleOrDefault(g => g.UserId == grade.StudentId)
    };
    gradeResponseDTO.Student.PasswordHash = null;
    gradeResponseDTO.Lecturer.PasswordHash = null;
    return gradeResponseDTO;
}

2 references
public List<GradeResponseDTO> GradeListToDTO(List<Grade> grades)
{
    List<User> query = _context.Users.ToList();
    List<GradeResponseDTO> gradeList = new List<GradeResponseDTO>();
    foreach (var grade in grades) {
        GradeResponseDTO gradeResponseDTO = new GradeResponseDTO()
        {
            Grade = grade,
            Lecturer = query.SingleOrDefault(g => g.UserId == grade.LecturerId),
            Student = query.SingleOrDefault(g => g.UserId == grade.StudentId)
        };
        gradeResponseDTO.Student.PasswordHash = null;
        gradeResponseDTO.Lecturer.PasswordHash = null;
        gradeList.Add(gradeResponseDTO);
    }
    return gradeList;
}
```

```

2 references
public List<GradeResponseDTO> GetByStudentId(int id)
{
    List<Grade> query = _context.Grades.ToList();

    List<Grade> grades = query.Where(g => g.StudentId == id).ToList();

    List<GradeResponseDTO> gradeList = new List<GradeResponseDTO>();

    gradeList = GradeListToDTO(grades);

    return (gradeList);
}

2 references
public List<GradeResponseDTO> GetByLecturerId(int id)
{
    List<Grade> query = _context.Grades.ToList();

    List<Grade> grades = query.Where(g => g.LecturerId == id).ToList();

    List<GradeResponseDTO> gradeList = new List<GradeResponseDTO>();

    gradeList = GradeListToDTO(grades);

    return (gradeList);
}

```

Buraya sorgu atılırken kullanıcı tipine göre kendi id'si ile sorgu atar. Ona göre

```

namespace StudentManagementSystemAPI.Models.DTO
{
    17 references
    public class GradeResponseDTO
    {
        2 references
        public Grade Grade { get; set; }

        4 references
        public User Lecturer { get; set; }

        4 references
        public User Student { get; set; }
    }
}

```

yine bir DTO

döner.

```

SqlServerModelBuilderExtensions.UseIdentityColumns(modelBuilder);

modelBuilder.Entity("StudentManagementSystemAPI.Models.Grade", b =>
{
    b.Property<int>("GradeId")
        .ValueGeneratedOnAdd()
        .HasColumnType("int");

    SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("GradeId"));

    b.Property<int>("LecturerId")
        .HasColumnType("int");

    b.Property<string>("LessonDate")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("LessonName")
        .HasColumnType("nvarchar(max)");

    b.Property<int>("Score")
        .HasColumnType("int");

    b.Property<int>("StudentId")
        .HasColumnType("int");

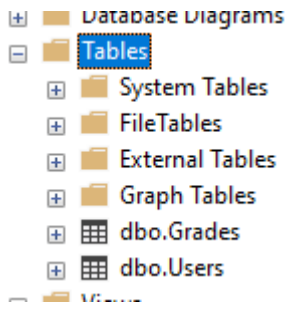
    b.HasKey("GradeId");

    b.ToTable("Grades");

    b.HasData(

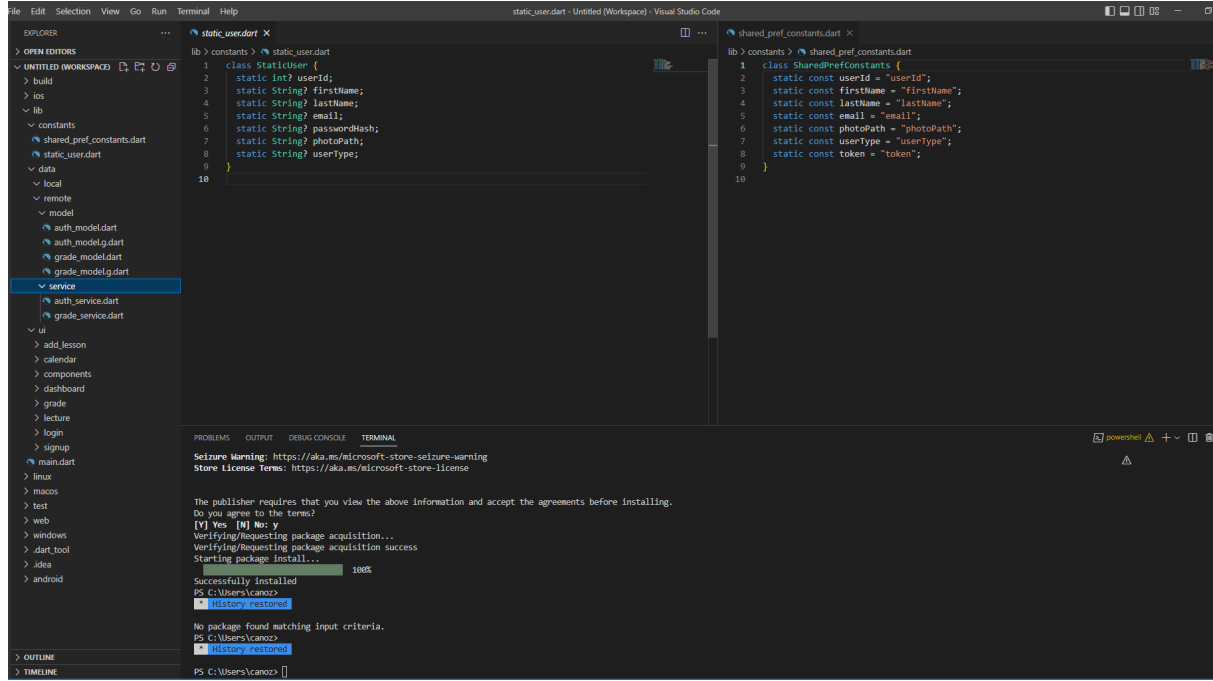
```

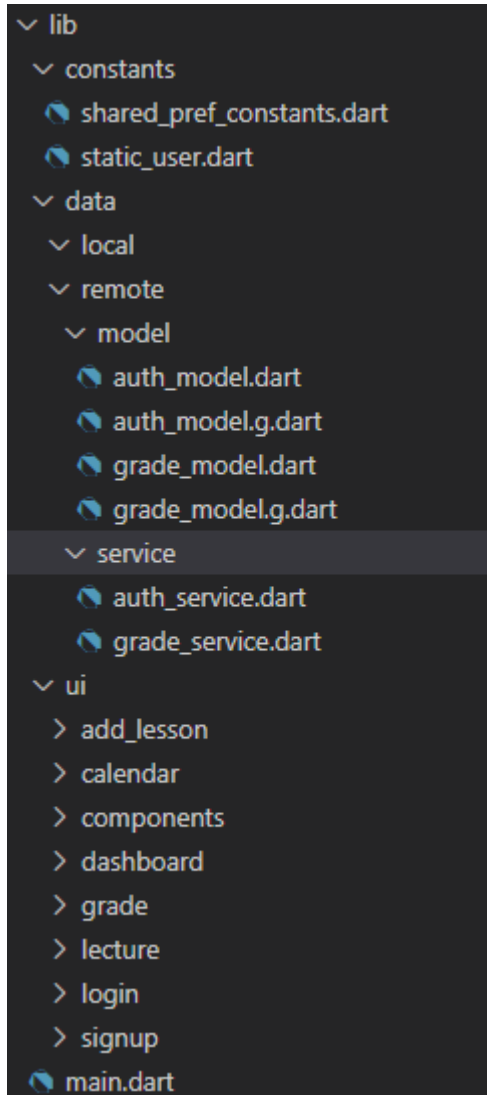
Database oluşturulurken Code-First kullanıldığı için migration yapılmıştır. Bu sayede



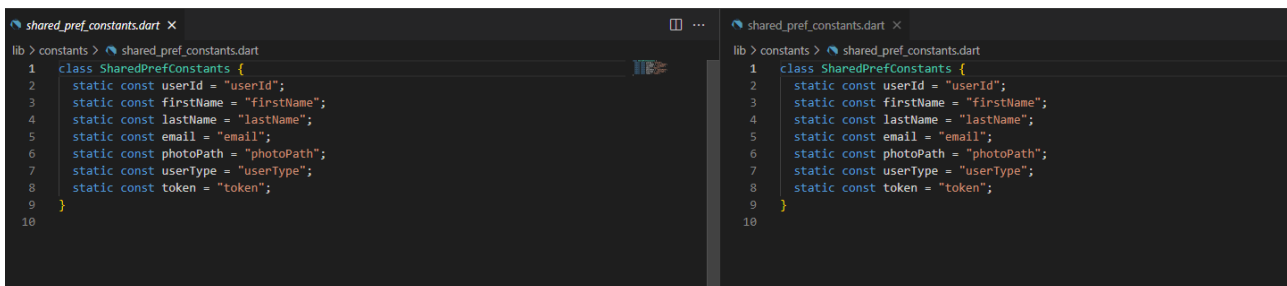
Tabloları oluşturulmuştur.

Web Uygulamasının Front Kısımında flutter kullanılmıştır.





API'den veri çekmek için servisler yazılmıştır. Bunları modellere mapleyip ui kısmında gösterilmiştir.



Giriş yapıldıktan sonra gelen user sessionu shared_pref ve static_user olarak tutulmuştur.

```
auth_service.dart X
lib > data > remote > service > auth_service.dart
1 import 'package:dio/dio.dart';
2 import 'package:student_management_system_app/data/remote/model/auth_model.dart';
3
4 class AuthService {
5   static AuthService service = AuthService._();
6
7   AuthService._();
8
9   final String BASE_URL = "http://4665-88-244-38-68.ngrok.io/api/";
10  final String LOGIN = "login";
11
12  Dio dio = Dio();
13
14  Future<AuthModel> login(Map<String, String> loginModel) async {
15    final url = "$BASE_URL$LOGIN";
16
17    try {
18      Response response = await dio.post(url, data: loginModel);
19      AuthModel model = AuthModel.fromJson(response.data);
20      return model;
21    } on DioError catch (e) {}
22    print(e);
23  }
24
25  return null;
26
27 }
28

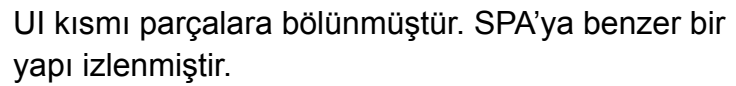
grade_service.dart
lib > data > remote > service > grade_service.dart
1 import 'package:dio/dio.dart';
2 import 'package:student_management_system_app/constants/static_user.dart';
3 import 'package:student_management_system_app/data/remote/model/grade_model.dart';
4
5 class GradeService {
6   static GradeService service = GradeService._();
7
8   GradeService._();
9
10  Dio dio = Dio();
11
12  final String BASE_URL = "http://4665-88-244-38-68.ngrok.io/api/";
13  final String GRADE = "grade";
14  final String USER = "user";
15
16  Future<List<GradeModel>> fetchAllGrades() async {
17    var url = "$BASE_URL$GRADE";
18    var id = StaticUser.userId == "student" ? "studentId" : "lecturerId";
19
20    try {
21      Response response = await dio.get(url, queryParameters: {id: StaticUser.userId});
22      List<GradeModel> grades = (response.data as List).map((e) => GradeModel.fromJson(e));
23      return grades;
24    } on DioError catch (e) {
25      print(e);
26      return List.empty();
27    }
28  }
29 }
```

Servisler şekildeki gibidir. Static base url olarak dışarıda test yapma amaçlı ngrok ile tünel açılmıştır. Kullanımı free version olduğu için uygulama tekrar çalıştığında endpoint değişecektir.

```
grade_model.dart X
lib > data > remote > model > grade_model.dart
1 // ignore_for_file: must_be_immutable
2
3 import 'package:equatable/equatable.dart';
4
5 part 'grade_model.g.dart';
6
7 class GradeModel extends Equatable {
8   Grade? grade;
9   Lecturer? lecturer;
10  Lecturer? student;
11
12  GradeModel(this.grade, this.lecturer, this.student);
13
14  GradeModel.fromJson(Map<String, dynamic> json) {
15    grade = json['grade'] != null ? Grade.fromJson(json['grade']) : null;
16    lecturer = json['lecturer'] != null ? Lecturer.fromJson(json['lecturer']) : null;
17    student = json['student'] != null ? Lecturer.fromJson(json['student']) : null;
18  }
19
20  @override
21  List<Object>? get props => [grade, lecturer, student];
22
23 }
24
25 class Grade extends Equatable {
26   int? gradeId;
27   String? lessonName;
28   int? lecturerId;
29   int? studentId;
30   int? score;
31 }
32

auth_model.dart X
lib > data > remote > model > auth_model.dart
1 import 'package:equatable/equatable.dart';
2 import 'package:json_annotation/json_annotation.dart';
3
4 part 'auth_model.g.dart';
5
6 @JsonSerializable(createToJson: false)
7 class AuthModel extends Equatable {
8   User? user;
9   String? token;
10
11  AuthModel(this.user, this.token);
12
13  factory AuthModel.fromJson(Map<String, dynamic> json) {
14    return _AuthModelFromJson(json);
15  }
16
17  @override
18  List<Object>? get props => [user, token];
19
20 }
21
22 @JsonSerializable(createToJson: false)
23 class User extends Equatable {
24   int? userId;
25   String? firstName;
26   String? lastName;
27   String? email;
28   String? passwordHash;
29   String? photoPath;
30   String? userType;
31 }
```

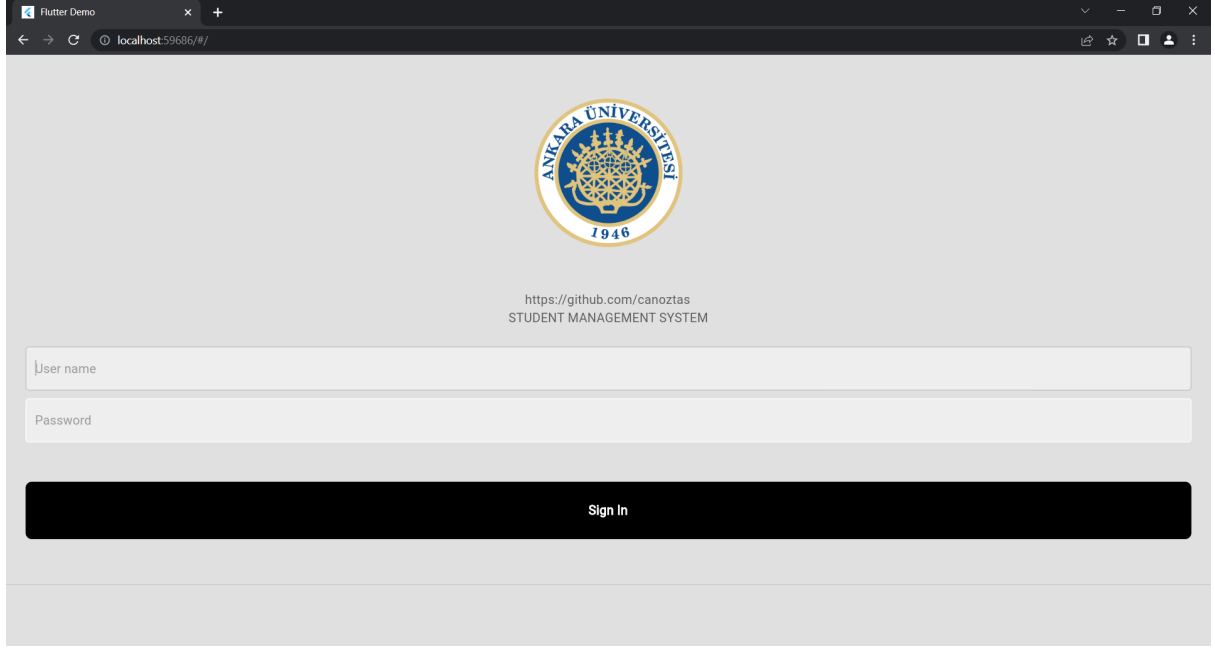
API'den dönen jsonlar modellere parse edilmiştir.



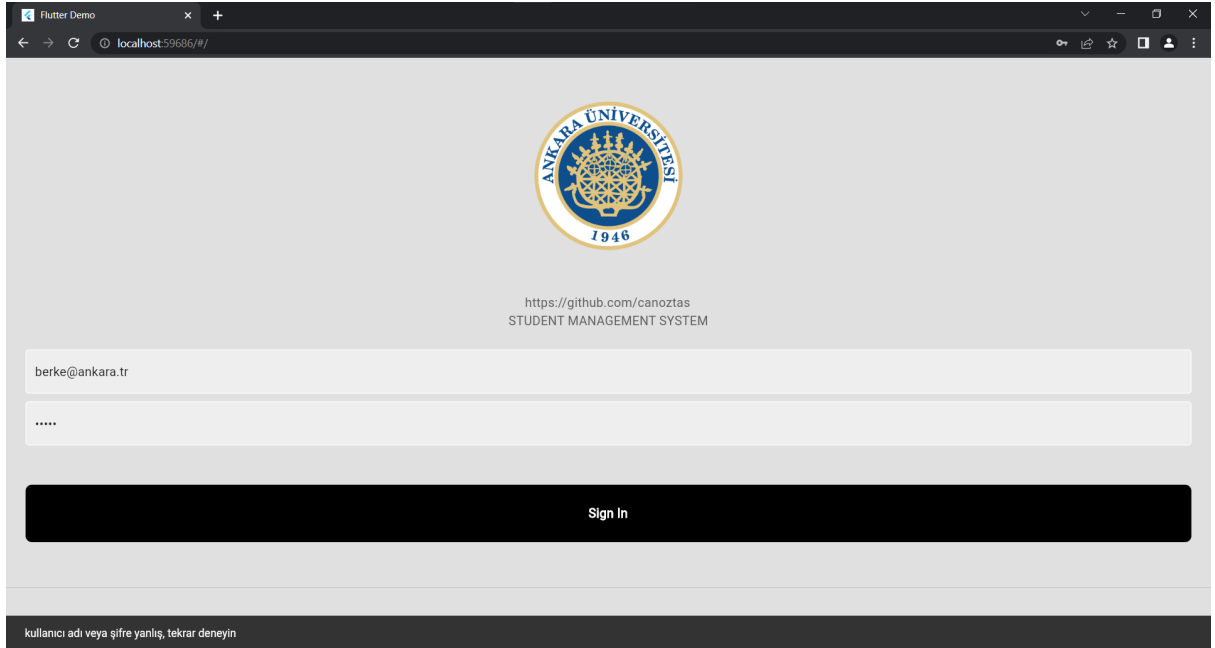
```
lib > ui > add_lesson.dart
80 mainAxisAlignment: MainAxisAlignment.spaceBetween,
81 crossAxisAlignment: CrossAxisAlignment.start,
82 children: [
83   Expanded(
84     child: Text(
85       model.grade?.lessonName ?? "",
86       style: Theme.of(context).textTheme.labelLarge,
87     ),
88   ),
89   Expanded(
90     child: ListTile(
91       title: Text(
92         "${model.student?.firstName} ${model.student?.lastName}",
93         style: Theme.of(context).textTheme.labelLarge,
94       ),
95       subtitle: Text(
96         "${model.student?.email}",
97         style: Theme.of(context).textTheme.labelMedium,
98       ),
99     ),
100   ),
101   Expanded(
102     child: TextField(controller: controller),
103   ),
104 ],
105 ),
106 ],
107 ),
108
```

```
lib > ui > calendar.dart
41 buildLesson(model: model, grade: grade, elevation: 10, title:
42   ),
43   Expanded(
44     child: SizedBox(
45       height: 70,
46       child: Card(color: Colors.grey[600], elevation: 10, child:
47   ),
48   Expanded(
49     child: SizedBox(
50       height: 70,
51       child: Card(color: Colors.grey[600], elevation: 10, child:
52   ),
53   Expanded(
54     child: SizedBox(
55       height: 70,
56       child: Card(color: Colors.grey[600], elevation: 10, child:
57   ),
58   Expanded(
59     child: SizedBox(
60       height: 70,
61       child: Card(color: Colors.grey[600], elevation: 10, child:
62   ),
63   ),
64   Row(
65     children: [
66       Expanded(
67         child: SizedBox(height: 400, child: buildLesson(model, "
68
```


Uygulama içi görüntüler:




Login ekranı



Login ekranı auth olmazsa hata veriyor

Flutter Demo

localhost:59686/#/



https://github.com/canoztas
STUDENT MANAGEMENT SYSTEM

enverbagci@ankara.tr

.

Sign In

Öğretmen tipi kullanıcı giriş yapıyor

Flutter Demo


localhost:59686/#/

Dashboard

OBS ANKARA

Enver Bagci

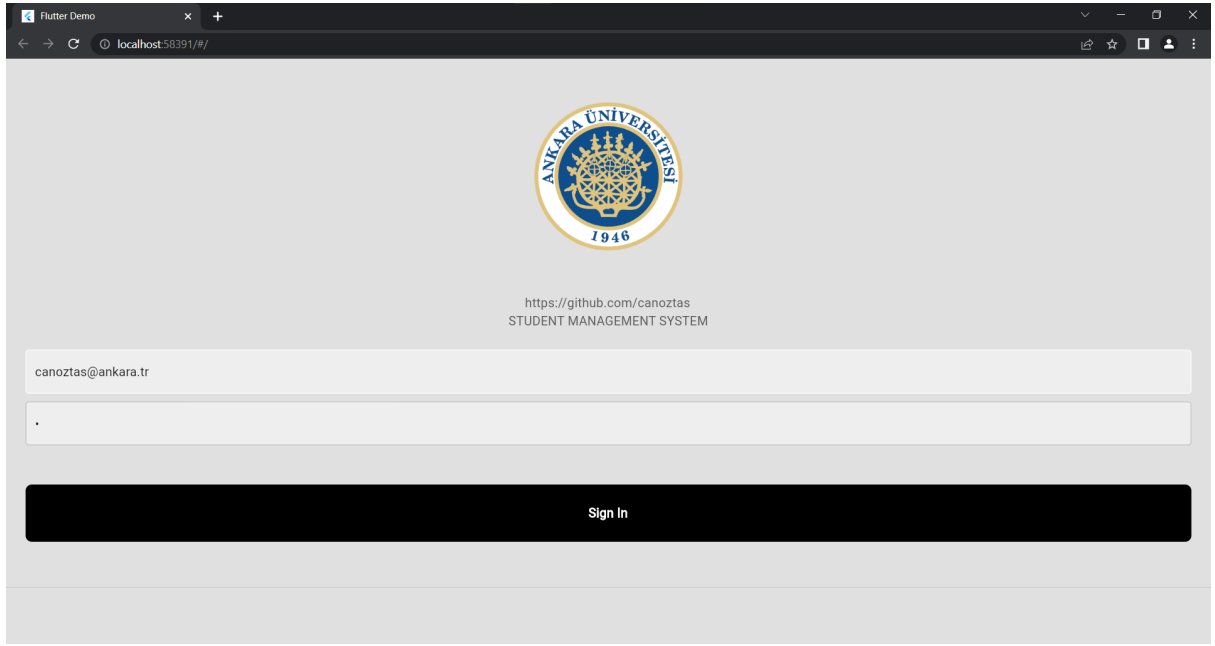
Mail: enverbagci@ankara.tr /Type: lecturer



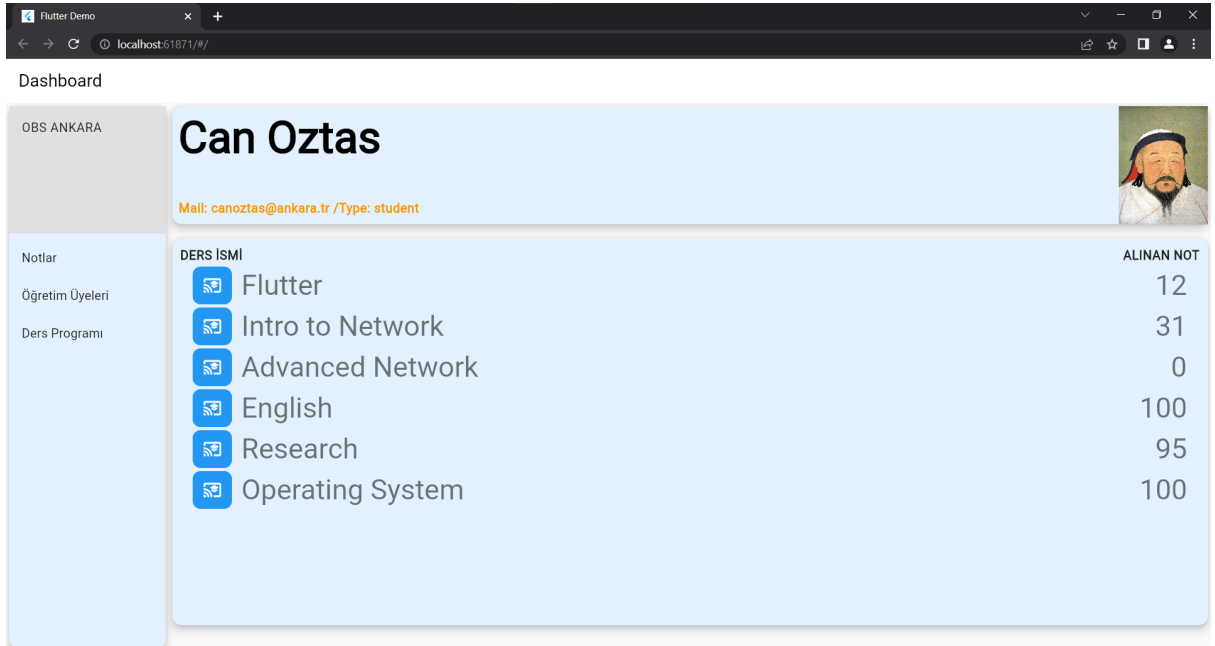
DERS ISMI	ÖĞRENCİ ISMI	ALINAN NOT
Flutter	Can Oztas canoztas@ankara.tr	12
Intro to Network	Jon Snow jon@snow.com	77
Intro to Network	Can Oztas canoztas@ankara.tr	31
Advanced Network	Can Oztas canoztas@ankara.tr	0
English	Can Oztas canoztas@ankara.tr	100
Research	Can Oztas canoztas@ankara.tr	95
Operating System	Can Oztas	31

KAYDET

Not düzenleme ekranı, öğretmen sahip olduğu tüm öğrencilerin notlarını düzenleyebiliyor. Yukarıda kullanıcı bilgileri yer alıyor



Öğrenci tipi kullanıcı giriş



Öğrenci notlarını görüyor, öğretmenden farklı olarak bazı endpointlere erişim yok.
(Not ekleme)

Flutter Demo

localhost:61871/#/

Dashboard

OBS ANKARA

Can Oztas

Mail: canoztas@ankara.tr /Type: student

Notlar

Öğretim Üyeleri

Ders Programı

İSİM SOYİSİM

Flutter

Intro to Network

Advanced Network

English

Research

ÖĞRETİM ÜYESİ

İsim: Enver Bagci

Mail: enverbagci@ankara.tr

İsim: Enver Bagci

Mail: enverbagci@ankara.tr

İsim: Jon Stark

Mail: jon@jon.com

İsim: Jon Stark

Mail: jon@jon.com

İsim: Enver Bagci

Sadece aldığı dersleri veren öğretmenleri görebiliyor.

Flutter Demo

localhost:61871/#/

Dashboard

OBS ANKARA

Can Oztas

Mail: canoztas@ankara.tr /Type: student

Notlar

Öğretim Üyeleri

Ders Programı

MONDAY

TUESDAY

WEDNESDAY

THURSDAY

FRIDAY

Advanced Network

Flutter

English

Research

Intro to Network

Operating System

Aldığı derslerin ders programı